Republic of Iraq
Ministry of Higher Education
and Scientific Research
University of Babylon
College of Engineering
Department of Electrical Engineering

Field Programmable Gate Array Resource Minimization of Polar Code Decoder Based on Successive Cancellation Algorithm

Submitted to the Council of the college of Engineering, University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master of Science (M.Sc.) in Electrical Engineering / Communications

By

Zubaidah Mahdi Shamer khudhair

Supervised by

Prof. Dr. Ahmed Abdulkadhim Hamad

2023A.D **1444 A.H**

بِسْمِ اللَّـهِ الرَّحْمَـٰنِ الرَّحِيمِ

﴿ هُوَ الَّذي جَعَلَ الشَّمسَ ضِياءً وَالقَمَرَ نورًا وَقَدَّرَهُ مَنـازِلَ لِتَعلَموا عَدَدَ السِّنينَ وَالحِسابَ ما خَلَقَ اللَّـهُ ذلِكَ إِلّا بِالحَقِّ يُفَصِّلُ الآياتِ لِقَومٍ يَعلَمونَ﴾

صدق الله العلي العظيم

سورة يونس ﴿5﴾

اقرار لجنة المناقشة

نحن اعضاء لجنة المناقشة, نشهد بأننا اطلعنا على رسالة الماجستير الموسومة **(تقليل مصادر مصفوفة البوابات المنطقية القابلة للبرمجة موقعيا الخاصه بالمرمز القطبي باستخدام خوارزمية الالغاء المتتالي)** وقد ناقشنا الطالبة **زبيده مهدي شمير خضير** في محتوياتها وفيما له علاقة بها , نؤيد انها جديره بالقبول لنيل درجة الماجستير علوم في الهندسة الكهربائية / اتصالات .

رئيس اللجنة
التوقيع :
الاسم : أ.د. سعد سفاح حسون
التاريخ :

عضو اللجنة
التوقيع :
الاسم : أ.م.د. ايفان عبد الزهرة هاشم
التاريخ :

عضو اللجنة
التوقيع :
الاسم : أ.م.د. مصطفى رشيد اسماعيل
التاريخ :

عضو اللجنة (مشرفا)
التوقيع :
الاسم : أ.د. احمد عبد الكاظم حمد
التاريخ :

مصادقة رئيس القسم
التوقيع :
الاسم : أ.د. قيس كريم عمران
التاريخ :

مصادقة عميد الكلية
التوقيع :
الاسم : أ.د. حاتم هادي عبيد
التاريخ :

Supervisor’s certification

I certify that this thesis entitled ' **Field Programmable Gate Array Resource Minimization of Polar Code Decoder Based on Successive Cancellation Algorithm** ' was prepared by **Zubaidah Mahdi Shamer khudhair** under my supervision at the as part of the prerequisites for a master's degree in engineering, department of electrical engineering, college of engineering, university of Babylon /electrical engineering / Communications.

Supervisor

Signature:

Name**: Prof. Dr. Ahmed Abdulkadhim Hamad**

Date: / / 2023

In light of the foregoing proposal, I am submitting this thesis to the Examination Committee for review.

Head of Electrical Department

Signature:

Name: Prof. Dr. Qais Kareem Omran

Date: / / 2023

Examining certificate from the committee

We confirm that we have read and understood the thesis named ' **Field Programmable Gate Array Resource Minimization of Polar Code Decoder Based on Successive Cancellation Algorithm** ' and as an examining committee, examined the student **Zubaidah Mahdi Shamer khudhair** in its substance, and that it, in our judgment, fits the requirements for a master's thesis engineering/electrical engineering / Communications.

Signature:	Signature:	Signature:
Name: Prof.	Name: Asst. Prof.	Name: Asst. Prof.
Dr. Saad Saffah Hassoon	Dr. Ivan Abdul-Zahra Hashim	Dr. Mustafa Rashid Ismael
(chairman)	(member)	(member)
Date: / / 2023	Date: / / 2023	Date: / / 2023

Signature:	Signature:
Name: Prof.	Name: Prof.
Dr. Ahmed Abdulkadhim Hamad	Dr. Qais Kareem Omran
(supervisor)	(Head of Electrical Department)
Date: / / 2023	Date: / / 2023

Signature:
Name: Prof.
Dr. Hatem Hadi Obeid
(Dean of College of Engineering)
Date: / / 2023

Dedication

To

my beloved mother and wonderful father; the origin of my success

my dearest husband and kids

my dearest sisters and brothers

I dedicate this work

Zubaidah Mahdi

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God Almighty for granting me success and giving me the capabilities that enabled me to finish this project successfully.

All thanks, appreciation and gratitude to my supervisor, who I am proud to be his student " **Prof. Dr. Ahmed Abdulkadhim Hamad** *" for all his support, guidance and valuable information as well as his trust in me during this work.*

I would like to thank all members of the (College of Engineering Electrical Engineering Department at the Babylon University) who lit a candle in the paths of our knowledge and gave the outcome of their ideas to enlighten our path.

Special thanks and appreciation to all my friends who gave me so much help and support.

Finally, I would like to extend my sincere thanks and appreciation to everyone who helped me achieve this message, even with one piece of advice or a word of support.

Abstract

The communication system is a model system that delineates a sophisticated exchange between two pivotal stations: the transmitter and the receiver. As signals or information pass from their origin to their ultimate destination, they navigate a complex channel imbued with numerous impairments, such as noise, attenuation, and distortion. Therefore, error-correcting codes are used for controlling errors in data when transmitted over unreliable or noisy communication channel.

Polar codes represent an emerging class of error-correcting codes with power to approach the capacity of a discrete memoryless channel. Polar code is chosen for the control channel in the 5G, new radio (NR), because it can fix errors well even at a low signal-to-noise power ratio. It has proven that the performance can get close to the Shannon limit for large frame lengths. The third-generation partnership project (3GPP) has chosen polar codes as the channel coding scheme for the Enhanced Mobile Broad Band (EMBB) control channels.

This work proposes an FPGA implementation of Polar codes that use successive cancellation (SC) algorithm. The SC decoder is the conventional decoder of polar code which its input is log-likelihood ratio (LLR) and its output is (binary). The SC decoding algorithm of Polar Code is improved and optimized to make it more practical. It has been shown that using less number of bits to represent the LLR values in the decoder has a significant effect on the used silicon area and latency without a significant effect on the performance.

In addition to the native C++ standard data types created on the 8-bit boundary, Vivado high-level synthesis (HLS) provides bit-accurate types that allow any arbitrary bit-width to be specified. In light of this, it also provides a comparison of building the successive cancellation (SC) polar code decoder utilizing various

arbitrary-precision fixed data types rather than the more expensive float representation to reduce processing time and utilized space.

The FPGA device (Kintex-7, Xilinx part number XC7K325T-2FFG900C) was utilized with various parallelism directives to ensure the targeted initiation interval and silicon area. This was made possible thanks to the FPGA device's high degree of flexibility in the designing and implementing of prototype systems. Different directives provided by HLS, like, pipelining, unrolling, and array partitioning, are utilized to achieve a tradeoff between performance (latency and throughput) and silicon area. It has been proved that using directives has no impact on the BER performance but reduces decoding time. Different simulation tests are carried out over Additive White Gaussian Noise (AWGN) channel to prove the effectiveness of the proposed scheme.

The results show a reduced latency of about 76% and a noticeable decrease in logic elements and DSP units (DSP48E) by about 50% and 86%, respectively. The hardware run of different polar coded systems with an arbitrary precision instead of float data type reveals a shallow degradation in performance.

Table of Contents

List of Tables

List of Figure

List of Abbreviation

Abbreviation	Definition
2b-SC	two bits successive cancellation
APP	A posterior probability
Apx-CUs	Approximated computing units
AWGN	Additive White Gaussian Noise
B-DMC	Binary-input discrete memoryless channel
BER	Bit Error Rate
BI-DMS	Binary input discrete memoryless symmetric
BLER	Block Error Rate
BP	Belief *Propagation*
BPSK	Binary phase-shift keying
CD	compact disc
CLBs	Configurable logic blocks
DSP	Digital signal processor
ECC	Error correction codes
EDA	Electronic Design Automation
EMBB	Enhanced Mobile Broadband
FER	Frame Error Rate
FF	Flip-Flops
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HLS	High Level Synthesis
IOBs	Input/Output blocks
LDPC	Low Density Parity Check
LFSR	Linear Feedback Shift Register
LiDAR	Light Detection and Ranging,
LLR	Log likelihood ratio
Log-MAP	Log Maximum A Posteriori Probability
LTE	Long Term Evolution
LUT	Look-Up Tables
MAP	Maximum A posteriori Probability
MIG	Memory Interface Generator
ML	Maximum Likelihood
NR	New Radio
RAM	Random-access memory
RTL	Register Transfer Level
SC	Successive cancellation
SCAN	Soft Cancellation
SCL	Successive Cancellation List

SNR	Noise Power Ratio
SRAM	Static Random-access memory
SSC	simplified successive cancellation
FSSC	fast simplified successive cancellation
TPSC	Two-phase successive cancellation
VHDL	VHSIC-Hardware Description Language

List of Symbols

Symbol	Definition
$\hat{c}_i$	the decoder's decision for i-th coded bit
$\hat{u}_i$	decoded bit
2^K	(N, K, R) block code
$\mathcal{F}^{\otimes n}$	nth Kronecker power of $\mathcal{F}$
B_N	bit reversal transposition matrix
$L_{N-1}^{(e)}$	outbound message LLR
$L_k^{(e)}$	Extrinsic LLR
$L_k^{(full)}$	full LLRs
$L_k^{(i)}$	Intrinsic LLR
$\acute{c}$	received message
u_0^{N-1}	data bit sequence
x_0^{N-1}	equivalent encoded bit sequence
σ^2	noise-variance
$\otimes n$	nth Kronecker power
AP_SAT	Saturation
AP_TRN	Truncation to minus infinity
C	codeword
dB	Decibel
h(.)	hard decision (data)
I	number of fractional bits
L	LLR value
N	Polar code length, noise power
Pr	probability of error
Q and O	represent the Quantization and Overflow Modes respectively which can take different forms
r	Code Rate, received value from the channel
S	power of a signal
E_b	broadcast energy per bit
$/N_oE_b$	Energy per bit over noise spectral density
G	generation matrix of a polar code
$I(\mathcal{W})$	symmetric channel capacity
N_o	Noise spectral density
R	rate of data transmitted
$Z(\mathcal{W})$	Bhattacharyya parameter
m	source Message

$p(n)$	Probability of density
$\mathcal{W}$	Bandwidth, B-DMC
$\mathcal{W}(y \mid x)$	transition probability
$\mathcal{X}$	input alphabets
$\mathcal{Y}$	output alphabets

Chapter One

Introduction

1.1 Background

The data in the communication systems were interpreted as a series of binary bits. These bits are modulated and transmitted via a communication channel. A communication channel is a medium where the information is transmitted from the sender to the receiver. This signal is corrupted by the noise, distortion, and interference introduced by the channel; due to these impairments, the received signal may not be the same as the original signal that was transmitted [1].

The rapid development of wireless communication systems in recent years has contributed to the increased efficiency of data transmission in today's world. The channel coding application in these systems makes it an essential component of the design of those systems. It is beneficial because it helps lessen the impact of the transmitted signal being contaminated by noise. Because of this, it has become an indispensable unit for all contemporary communication standards.

It has been demonstrated that Polar codes, which Arikan first developed in 2009, may achieve the memoryless channels' capacity [2]. Polar codes have several advantages as opposed to turbo codes and low-density parity-check (LDPC) code implemented in 4G networks. These advantages include less complexity of encoding and decoding schemes, low latency as a result of the parallel implementation of some decoding methods, and the absence of an error floor at a high signal-to-noise ratio (SNR) [3,4]. Recent events have resulted in the selection of polar codes for use in Enhanced Mobile Broadband (EMBB) control channels in advance for the 5G NR

(New Radio) interface [5]. Polar Codes are often created for Additive White Gaussian Noise (AWGN) channels. However, they may not be adequate when used with frequency-selective channels, which is typically the situation in practice. It is more feasible to change the receiver as opposed to the Polar encoder, even though certain investigations are going on for the construction of Polar Code in memory channels. [6]–[8]

1.2 Channel coding

Many techniques are discovered to mitigate the effects of distortion caused by the channel. In digital communication systems, Channel coding is used to mitigate the effect of noise and interference that corrupt the received data and decrease bit errors. Channel coding is performed primarily by the deliberate addition of redundant bits into the stream of information that is being transmitted. The binary message sequence that is received by the channel encoder might have come from the output of a source encoder or it could have come straight from the source itself [9].

Systematic redundancy is introduced into the data stream by the channel encoder, which does so by adding bits to the message bits in such a way as to make it easier for the receiver to detect and/or correct bit errors in the original binary message sequence. This is accomplished by the channel encoder. The expense of implementing channel coding to protect data is an increase in bandwidth or minimize in data rate[9]. A simplified block diagram for a communication system applying error control code is depicted in the figure (1.1).

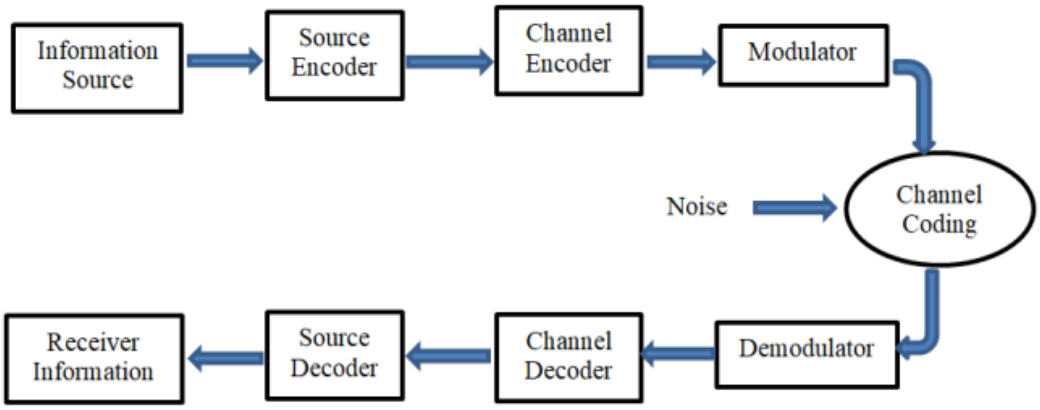


Figure (1.1): A Generalized System of Communication.

Shannon established the theoretical maximum rate of data transmission in 1940. The transferred information below this limit through a channel disturbed by additive-white-Gaussian-noise (AWGN) experiences a lower bit error rate. The maximum data rate is known as the Shannon capacity or the channel capacity. The formula for this capacity over the AWGN channel can express as follows [10]:

$$R < W\log_2\left(1+\frac{S}{N}\right) \text{ bits /sec} \qquad (1.1)$$

Where: R is the rate of data transmitted, W is the bandwidth, S is the power of a signal, and N is noise power. To achieve good performance and approach, Shannon limits concatenation codes were presented, where the codes are joined in a series or parallel scheme where each code can be decoded separately. The decoders of these codes can exchange information to increase reliability with reasonable complexity [11].

1.3 Polar Codes

Polar Codes build on the concept of channel polarization [2]. The code is in the first family to achieve maximum channel capacity on any symmetric binary-input discrete memoryless channel (B-DMC). It does this with small coding complexities

O(NlogN), where N represent the block length. Polar Codes are based on recursion where the original channel W_n is divided into virtual channels W_1 and W_2. Arikan proved that with enough division recursion, the cut-off rate would be higher on the virtual channel than on the original channel and that the virtual channels tend to have either high reliability or low reliability this is called code construction [12]. This means that the channels are either completely noisy or noiseless, where the noiseless channels should be chosen to transmit data on, and the noisy channel would assign to frozen bits [13]. It is sorted from worst to best in the reliability sequence, and the frozen bits are usually set to 0. Polar Codes were agreed upon to be used in the upcoming 5G - NR technology [14], replacing the now used Convolutional Codes in LTE. Encoding and decoding of polar code are explained in Figure (1.2).

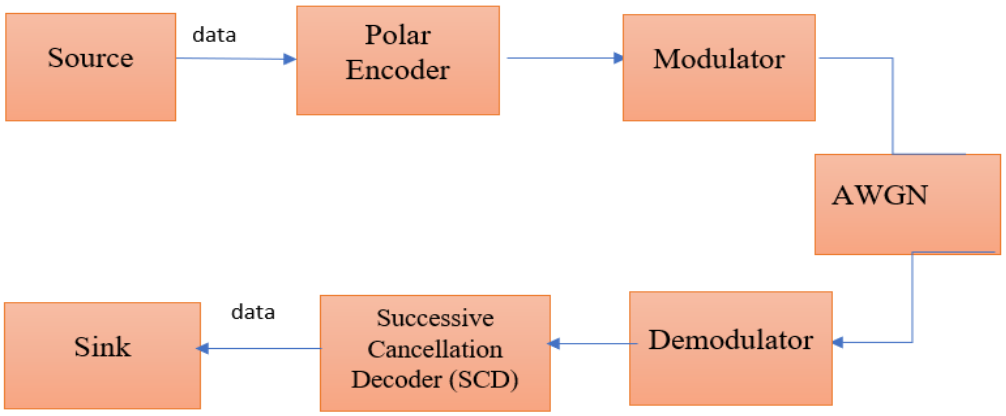


Figure (1.2): polar code encoding and decoding

The polar encoder is the first step in Polar Codes. It is responsible for polar-code construction for the data, and then the data will be modulated such that the receiver will know how to decode and verify the source message; in the encoder, the information bits are mapped according to a reliability sequence found in [14]. The

reliability sequence is built upon the channel polarization that the channel undergoes in the encoder.

A message that was encoded in Polar and sent over an AWGN channel is decoded by the Polar decoder when it is received, as seen in figure 1.2. The values of the messages that were received have signal offsets when compared to the messages that were sent. This is because the AWGN channel has noise on it. Polar Codes are constructed to have the capability of correcting these offsets and reconstructing the correct message from the data that has been received.

There are a variety of algorithms for decoding Polar encoded messages. The Successive Cancellation (SC) technique is the first decoding scheme for Polar Codes. Because of the structure of the Polar Codes, the values r that is received from the channel may be represented as a binary tree. This is possible because of how the Polar Codes are structured [13], as seen in figure (1.3) where L is LLR value, r is the received value from the channel. $\hat{u}_i$ Represent a decoded bit, and 0:s are frozen bits.

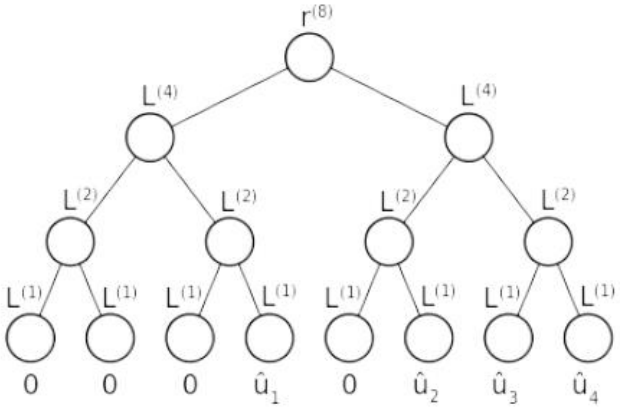


Figure (1.3): Successive Cancellation binary tree [13].

A common way of representing soft information for binary symbols is the log-likelihood ratio (LLR), which is identified as:

$$\text{LLR} = \log\left(\frac{\text{The likelihood of a bit being zero}}{\text{Likelihood of a bit being one}}\right)$$

The sign of the LLR describes the decision whether the bit is zero or one, whereas its magnitude corresponds to the reliability of this decision. The higher the magnitude is, the more confident the decoder is about its decision. For instance, if the LLR is positive, it means the decoder is more optimistic about the bit being zero rather than one. In the extreme case when this LLR is $+\infty$, the decoder is absolutely sure that the bit is zero. If the decoder decides only with absolute belief delivering only $+\infty$ and $-\infty$ LLRs, it is called a hard-output decoder. The iterative exchange of this soft information gradually reduces the error rate resulting in the improved reliability of a polar code communication system [15].

1.4 Literature Review

Many researchers continue to enhance and integrate digital communication systems. The polar code is one of the codes that researchers have worked on for its importance in communication systems and other applications. The literature survey for associated research in this field is as follows:

Leroux suggested using a semi-parallel framework for the execution of sequential cancellation decoding, which resulted in a processing complexity that was significantly reduced. This decoder has the potential to not only increase efficiency in the use of resources but also reduce the typical amount of time required for decoding [16].

There are two advantages to the modification presented by Yuan, Bo, and Keshab K. Parhi. Before anything else, the final stage of the SC algorithm drastically decreases the critical path and hardware complexity. Second, decoding can occur in two bits

rather than one bit. That's why the new decoder, called a 2b-SC decoder, can minimize latency from (2n-2) to (1.5n -2), where n is the number of steps in polar code structure and is equal to $\log_2(N)$, The 2b-SC-Precomputation decoder is 25% faster than the previous least-latency SC decoder. The suggested (1024, 512) 2b-SC-Precomputation decoder has been shown through simulation to provide at least a four times improvement in throughput and a forty percent improvement in hardware efficiency [17].

In [18], Pamuk, Alptekin, and Erdal Arkan have come up with an idea for a decoder architecture for polar codes called the two-phase successive cancellation (TPSC). This architecture makes use of the array-code aspect of polar codes by partitioning the decoding of a length-N polar code into a series of length-N decoding cycles. This allows the architecture to take full advantage of the array-code feature of polar codes. Every iteration of the decoding process is divided into two phases: the first phase decodes the code array in the direction of the columns, and the second phase decodes it in the direction of the rows. According to the findings, the suggested scheme is easier to understand and implement, makes less use of memory while yet achieving higher throughput, and has a clock frequency that is less dependent on changes in the length of the code.

Giard, Pascal, et al show in [19] how the low-complexity decoding algorithm can be improved to better accommodate low-rate codes. Dedicated hardware is added to efficiently decode new constituent codes. A polar decoder for a (1024, 512) code is implemented on two different FPGAs. It has 25% lower latency over the previous work and a coded throughput of 436 Mbps and 638 Mbps on the Xilinx Virtex 6 and Altera Stratix IV FPGAs, respectively.

In [20], Dizdar, Onur, and Erdal Arıkan suggested a high-throughput energyefficient Successive Cancellation (SC) decoder architecture for polar codes based on combinational logic. The proposed combinational architecture operates at relatively low clock frequencies compared to sequential circuits, but takes advantage of the high degree of parallelism inherent in such architectures to provide a favorable tradeoff between throughput and energy efficiency at short to medium block lengths. At longer block lengths, this architecture proposes a hybrid-logic SC decoder that combines the advantageous aspects of the combinational decoder with the low-complexity nature of sequential-logic decoders.

Huang, Y., Cai, Y., Jing, M., Han, J., Fan, Y., & Zeng, X. presented in [21] modification of successive cancellation (SC). They transform data from float point to fixed point, which results in a reduction of LUT resources of 63.5 percent, and they enlarge the recursion in the SC decoding algorithm in accordance with the rules of the SC decoding algorithm. In addition to this, they use pipeline in the for loop, which minimizes cycle time by 29%.

Chen, Y., Xia, Z. W., Tang, L. Y., Wan, G. C., & Tong, M. S. in [22] improved and optimized the SC decoding algorithm to strengthen enhance Polar Code's performance even more. The performance of the Block Error Rate (BLER) and the Bit Error Rate (BER) is also significantly enhanced at the same time. since, the FPGA is capable of processing data at a fast speed, doing computations in parallel, and being entirely flexible in its reconfiguration. As a result, they simplified the SC decoding method to make it compatible with the FPGA implementation.

Delomier, Y., Gal, B. L., Crenne, J., and Jego, C. [23] offer a model-based design process that may be used to develop efficient hardware SC polar code decoders. On

a device built with Xilinx Virtex-7 and an Altera Stratix IV, it is possible to reach decoding throughputs that are more than 300 Mbps.

1.5 Problem statement

In communication system the signal has been corrupted by the noise, distortion, and interference introduced by the channel; as a result of these problems, the received signal may differ from the original signal that was transmitted. Polar code which is a linear block error-correcting code is suggested to provide efficient communication with low latency and resources and high reliability. SC polar code decoder is implemented with the help of the VIVADO HLS program.

1.6 Aims and Objectives of the Work

An Outline of the Work's Aims and Objectives is to:

A - improve the performance of polar coded signals over the AWGN channel. This is achieved by utilizing the FPGA device (Kintex-7, Xilinx part number XC7K325T-2FFG900C) to implement the polar encoder and decoder.

B - reduce the time-to-market by using a high-level language (C++) in the design of the various parts comprising the digital communication system under test using the Vivado HLS platform.

C - make a tradeoff between latency and utilized FPGA resources using different directives (e.g., pipelining and loop unrolling) offered by Vivado HLS.

D - Applying a different fixed-point data type rather than the more expensive float type to represent the LLR information processed by successive cancellation (SC)

decoders may help reduce the latency, an average number of needed iterations, silicon area and average decoding time.

1.7 Thesis Layout

Chapter One introduces the basic communication systems, Channel coding, polar code, Literature Review, and the aims of the work.

Chapter Two introduces the Single Parity-Check Codes and Repetition Codes and how they are used together to build polar code and channel Polarization which is the key idea behind Polar Codes, polar code construction, and it introduces the structure of polar encoder and successive cancellation (SC) decoder. This chapter also explains the architecture of the target Field-Programmable-Gate-Array (FPGA), Common Applications of FPGAs, and a brief comparison with Microprocessor.

Chapter Three presents various suggested methods for improving the BER and frame error rate (FER) performance, reducing of average estimating decoding time, and the utilized resources in the designing and implementation of the polar encoder and decoder.

Chapter four: Different computer simulation tests are carried out using Microsoft Visual Studio and High-Level-Synthesis (HLS) to reveal the effect of code rates, and code lengths on the polar code performance. Different comparisons between floating point and fixed point with different accuracy precision are carried out. The counterpart FPGA implementation using Kintex 7 series is also presented. In this work, the Genesys 2 board provided by Digilent Company is used as the hardware platform for all practical tests.

Chapter Five: The Conclusion is presented, and some Future Works directions.

Chapter Two

Theoretical Background on Polar Code

2.1 Introduction

This chapter provides the necessary background required on polar code for the remainder of the thesis. Introduction to block codes and a description of two important types of these codes, namely the single parity-check and repetition codes. Next is an introduction to polar codes are presented, their factor graph construction, and how this factor graph relates to single parity-check and repetition codes are discussed. Later, the original successive cancellation (SC) decoder and the channel polarization of the polar code are discussed [1]. The general structure of the FPGA device and its application are presented. A comparison between FPGA and microprocessors is discussed. In the end, a comparison between High-Level Synthesis (HLS) and (VHDL) is discussed.

2.2 Coding Theory

When information is sent across a channel, it may be corrupted by noise in the channel. Coding theory studies how one detects or even corrects errors that occur due to noise. Error-correcting codes protect information from distortion caused by noise. Figure (2.1) demonstrates how message m is encoded to be sent across a noisy channel, and the received message $\acute{c}$ is decoded by the receiver. Applications of the coding theory include satellite communication, mobile, flash memories, CDs, and much more. [24,25]

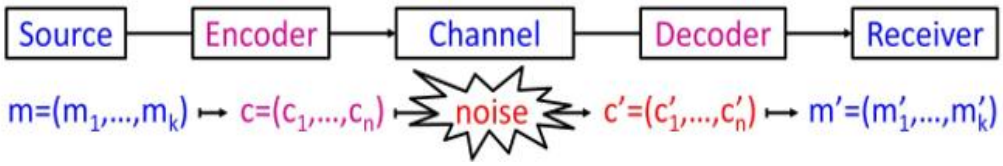


Figure (2.1): Basic encoder and decoder.

2.2.1 Block code:

A block code breaks the message into smaller blocks and encodes each block separately. The code is the set of all possible codewords. Figure (2.2) shows the typical diagram for block encoding. The large message is divided into chunks of K bits, and the encoder encodes each chunk separately into codewords of length N. The resulting set of 2^K codewords is called the (N, K, R) block code, where R equal to K/N is called the code rate and quantifies the fraction of information bits carried by each coded bit [26]. Two important block codes are single parity-check codes and repetition codes. Both of these codes are the building blocks of more powerful codes, such as LDPC codes.

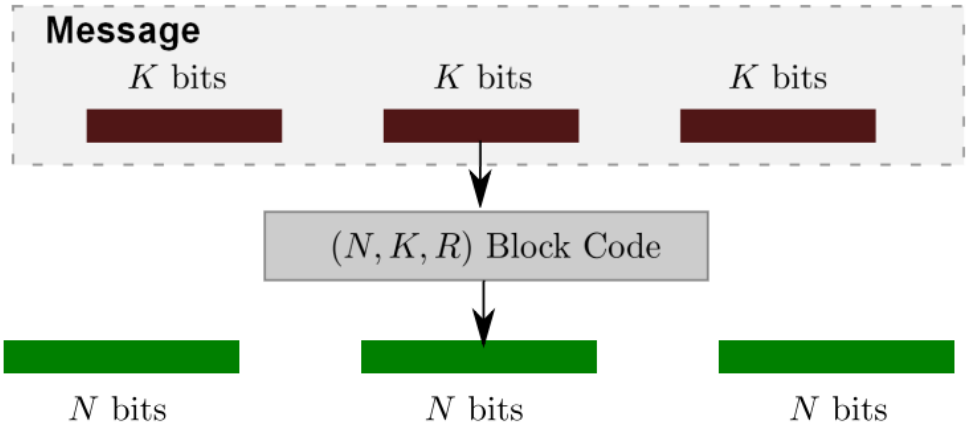


Figure (2.2): block code.

2.2.1.1 Single Parity-Check Codes

Single parity-check codes are (N, N − 1, (N – 1)/N) codes that append a single extra bit to the message. The extra bit is called the parity bit, as its value is the binary XOR of all the message bits. Suppose we want to transmit a binary message m = [1 0] using a (3, 2, 2/3) single parity-check code [15]. Figure (2.2) show that the message is divided into blocks of K bits. Each block is then independently encoded to a block of N-coded bits using an encoder.

We first compute the parity-check bit 1 = 1 ⊕ 0 and append it to the message forming the codeword c = [1 0 $\boxed{1}$], where the bit in the box is the parity bit. Now, suppose we transmit codeword c = [1 0 1] on a binary-input discrete memoryless channel with conditional probability distribution p (y | c). Let the channel observation be vector $\mathbf{y} = [y_0 \quad y_1 \quad y_2]$. The task of the decoder is to take this observation y and decide what is transmitted. The decoder's decision depends on which parameter it optimizes, and one such parameter is the probability of error Pr $(\hat{c}_i \neq c_i)$, where $\hat{c}_i$is the decoder's decision for the i-th coded bit.

A common optimization metric is error probability, which can be lower bounded as follows [15], see appendix A

For an (N, N − 1, N − 1/N) code, the extrinsic LLR and full LLR of kth bit are given by

$$L_k^{(e)} = 2\tanh^{-1}\left(\prod_{j\neq k} \tanh\left(\frac{L_j^{(i)}}{2}\right)\right) \tag{2.1}$$

and

$$L_k^{(full)} = L_k^{(e)} + L_k^{(i)} \tag{2.2}$$

respectively.

Figure 2.3 shows the whole decision process as a message passing along the edges of a graph. The rectangle represents a parity-check node, whereas the circles represent variable nodes. The parity-check node represents the constraint that all its inputs should sum to zero. The figure shows that to compute the extrinsic LLR for the parity bit, we send intrinsic LLRs observed from the channel on the edges corresponding to variable nodes c_0 to c_{N-2}. The parity-check node takes all these messages from different variable nodes and computes the outbound message $L_{N-1}^{(e)}$. The same process can be repeated for all other bits to compute their extrinsic LLRs and, eventually their full LLRs for detection.

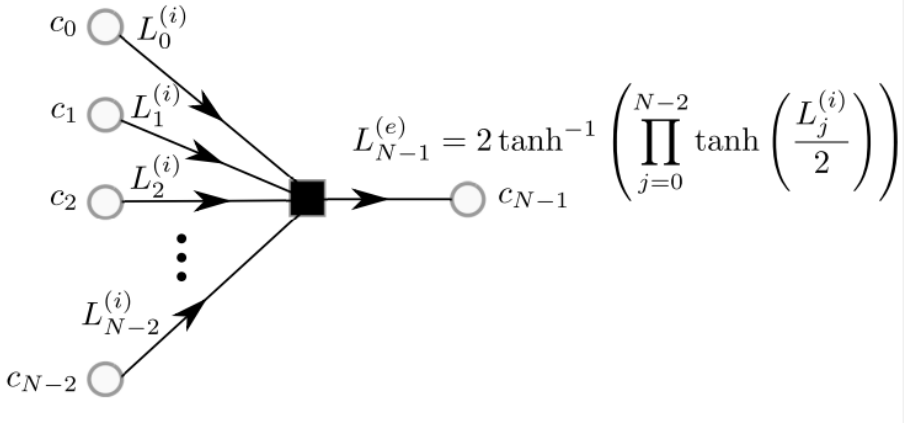


Figure (2.3): The parity-check node takes all the input messages in the form of intrinsic LLRs and produces the outbound message $L_{N-1}^{(e)}$.

2.2.1.2 Repetition Codes

An (N, 1, 1/N) repetition code repeats every message bit N times so that each message bit m_0 gets mapped to the codeword $\mathbf{c} = [m_0, m_1, \dots, m_{N-1}]$. Suppose we want to transmit a zero-bit using an (N, 1, 1/N) repetition code. This code takes the

message bit and transmits it N times, producing the following equality constraint between all these transmitted bits:

$$c_0 = c_1 = \cdots, = c_{N-1}.$$

Figure 2.4 shows that the decoding process for a repetition code can also be viewed as a message passing on a graph like parity-check codes [15]. The circle represents the variable node that puts the constraint that the bits corresponding to all the edges connected to it should be equal to each other. The left-hand image shows the trivial parity-check constraints of two edges. This two-edge parity-check is equivalent to an equality-check constraint meaning that the bits corresponding to the edges connected should be equal. The right-hand image is a rearranged form of the left-hand image and resembles the parity-check code graph.

Following a similar analysis as in Section 2.3.1, the MAP decision for the kth coded bit is given by

full LLR of kth bit are given by:

$$L_k^{(full)} \quad = L_k^{(e)} + L_k^{(i)} \tag{2.3}$$

In Figure 2.4, c_0 is a bit which is detected as all transmitted bits are equal and detecting this bit is enough to decode the message. To detect c_0We send all the intrinsic LLRs for $c_1, c_2, \ldots, c_{N-1}$ to the variable node of c_0. The variable node sums all the incoming messages and produces the extrinsic LLR $L_0^{(e)}$ That we add to intrinsic LLR $L_0^{(i)}$Calculating the full LLR. The sign of this full LLR decides about the message bit. It is easy to see that the full LLR for all the bits is equal to $\sum_{j=0}^{N-1} L_j^{(i)}$, highlighting once again that no matter which bit we decode, the resulting decision

for the message bit stays the same. However, note that extrinsic LLRs corresponding to different bits will generally differ.

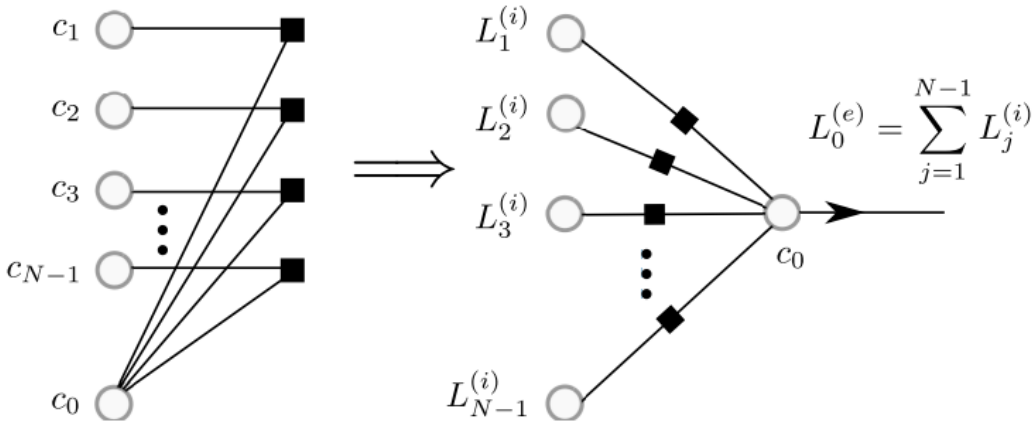


Figure (2.4): A (N, 1, 1/N) repetition code as a message passing on a graph.

2.2.2 Polar code

Polar codes were first built using Successive cancelation on the Bhattacharyya parameters of bit channels [27]. New to the coding scene, polar coding is an error-correcting technique introduced in [1]. The theoretical goal of providing an explicit code structure that is provably capacity attaining and implementable with low complexity drove the introduction of polar coding. Turbo and LDPC codes are two other families of codes that are practically implementable and known to have capacity-achieving performance yet require exact mathematical proofs that they genuinely reach capacity, except in some rare instances [28]. Alternatively, polar codes are amenable to rigorous mathematical examination. Furthermore, polar coding has been shown to be a flexible coding approach capable of reaching the information-theoretic limits in a variety of source and channel coding problems; a typical list of such work can be found in [29]-[34].

Let's think about the binary input discrete memoryless symmetric BI-DMS channel. Using a method known as channel polarization, N identically replicated independent BI-DMS channels can be transformed into N polarized channels (also known as bit channels). Since the channels are either extremely noisy or noiseless as N→ ∞, polarization has no effect on capacity. Then, by selecting to exclusively transmit over the excellent bit-channels, one can easily reach a transmission rate close to capacity. A bit error rate (BER) ranking algorithm is required to select K good channels from N when the block length N is limited and the rate R = K/N. For this purpose, we will refer to each code word of length N as having K bits of information [35]. For this reason, the process of selecting these bit channels to form a polar code is known as polar code creation.

2.2.2.1 Channel Polarization: The key idea behind Polar Codes

In his seminal work, Arıkan introduced the concept of channel polarization [2], where the probability of accurately detecting some fraction of the bits conveyed across a memoryless channel is improved by applying a polarizing transformation. Let $\mathcal{W}$ be a B-DMC, where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output alphabets, and the transition probability between them is $\mathcal{W}(y \mid x)$, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. To evaluate the quality of a channel, we can use two different metrics: the symmetric channel capacity $I(\mathcal{W})$and the Bhattacharyya parameter $Z(\mathcal{W})$, both of which are clear as follows: [36]

$$I(\mathcal{W}) = \sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} \frac{1}{2}\mathcal{W}(y \mid x)\log \frac{\mathcal{W}(y|x)}{\frac{1}{2}\mathcal{W}(y|0)+\frac{1}{2}\mathcal{W}(y|1)} \quad (2.4)$$

$$Z(\mathcal{W}) = \sum_{y\in\mathcal{Y}} \sqrt{\mathcal{W}(y \mid 0)\mathcal{W}(y \mid 1)} \quad (2.5)$$

The Bhattacharyya-parameter places upper constraint on likelihood of erroneous detection when using maximum likelihood ML to estimate a single received symbol y. The symmetric capacity is the highest rate by which reliable communications can take place, as concluded by $\mathcal{W}$. Both metrics have the values $\in$ [0, 1] when data is sent over a B-DMC, and they are linked by two inequalities [37]:

$$I(\mathcal{W}) \geq \log_2 \frac{2}{1+Z(\mathcal{W})} \tag{2.6}$$

$$I(\mathcal{W}) \leq \sqrt{1 - Z(\mathcal{W})^2} \tag{2.7}$$

From which it can be observed that $I(\mathcal{W}) \approx 1$ iff $Z(\mathcal{W}) \approx 0$ and $I(\mathcal{W}) \approx 0$ iff $Z(\mathcal{W}) \approx 1$; where $I(\mathcal{W})= 1$ denotes a perfect channel in which information may be transmitted without error, and $I(\mathcal{W})=0$ denotes a channel in which no transmission can occur (a completely-unreliable channel). Two values $(y_0,y_1) \in \mathcal{Y}^2$ are received (as shown in Figure (2.5a)) with the following mutual data values when two bits$(u_0,u_1) \in \mathcal{X}^2$ are communicated using two distinct instances of $\mathcal{W}$ or by utilizing the memoryless $\mathcal{W}$ twice :

$$I(Y_0, Y_1; U_0) = I(\mathcal{W}) = I(Y_0, Y_1; U_1) \tag{2.8}$$

Instead, the common information values between information besides the usual symbols improve if we change(u_0,u_1)into(x_0, x_1)so that: $x_0 = u_0 \oplus u_1$ also $x_1 = u_1$ (as in Figure (2.5b)):

$$I(Y_0, Y_1; U_0) \leq I(\mathcal{W}) \leq I(Y_0, Y_1; U_1) \tag{2.9}$$

Thus, the probability of making an accurate prediction of u_0 diminishes while that of u_1 rises. Proof of this inequality is presented in [2]. The polarizing transformation can be functional recursively to transform multiple instances (or uses) of $\mathcal{W}$, as

shown in Figure (2.5c) for eight instances of the channel. Estimating a symbol's u_i value accurately approaches either the probability of 1.0 (totally reliable) or 0.5 (completely unreliable) as the number of modified channels, N→∞, grows. Similarly, the proportion of bits that can be reliable is getting closer to the channel's maximum capacity, which means that the coding rate, R, is getting closer to the channel's maximum capacity, C. [38,39]

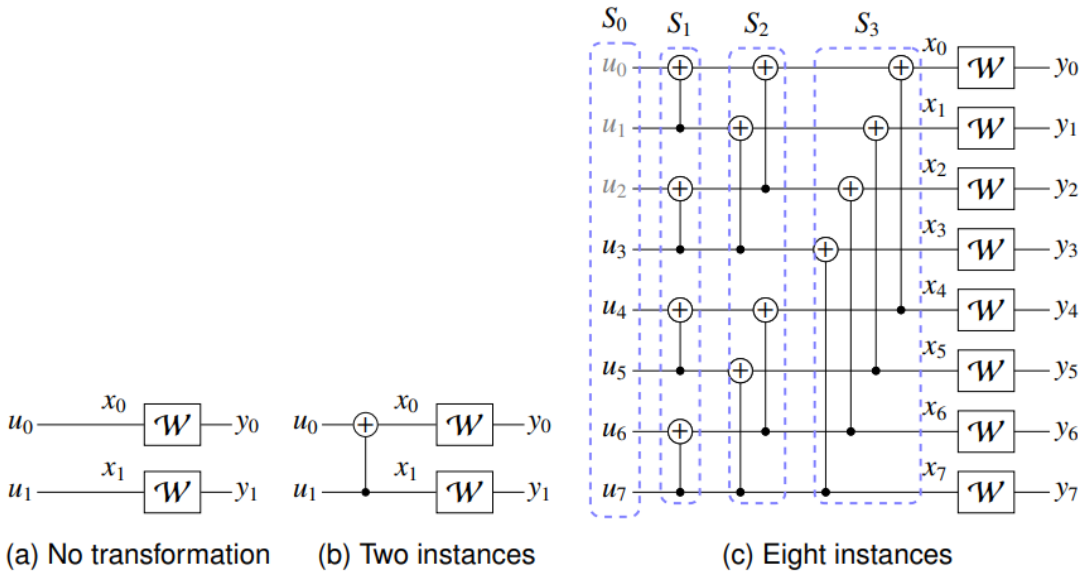


Figure (2.5): Polarizing transformation [36].

2.2.2.2 Encoding of Polar Codes

The generation matrix of a polar code is an N × N matrix that has the equation $G = B_N \mathcal{F}^{\otimes n}$. In this equation, N is equal to 2^n, B_N is the bit reversal transposition matrix, and $\mathcal{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. $\otimes n$ is the nth Kronecker power, $\mathcal{F}^{\otimes n} = \mathcal{F} \otimes \mathcal{F}^{\otimes(n-1)}$. If we define the data bit sequence $u_0^{N-1} = (u_0, u_1, \cdots, u_{N-1})$and the equivalent encoded bit sequence as $x_0^{N-1} = (x_0, x_1, \cdots, x_{N-1})$ therefore $x_0^{N-1} = u_0^{N-1} G$. The values of the encoded the bit sequence u_0^{N-1} are separated into two pairs: the data bits set A

contains K values, besides the frozen bits set A^c which is equal to N -K values. u_{A^c} representing the frozen bits, and their values come from A^c . Figure (2.6) displays the graph representing the encoding of a polar code by N equal to 8.

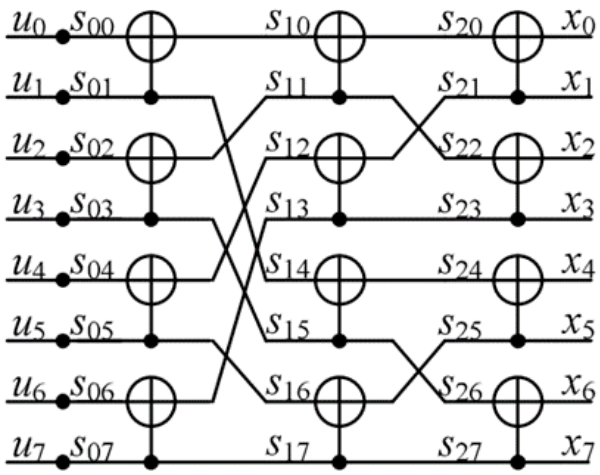


Figure (2.6): Polar encoder with N=8, R=1/2.

There are two methods for implementing a polar encoder. Matrix multiplication is the first method, and it has quadratic complexity. The complexity is reduced to O (logarithm of the number of butterflies) with butterfly recursive encoding (N log N).[40]

Having established this, the recursion that generates the generator matrix in [41] Kernel matrix of Arikan can be defined as $\mathcal{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. The n-th Kronecker product of $\mathcal{F}$ is indicated by the symbol $\mathcal{F}^{\otimes n}$, and its definition is as follows:

$$\mathcal{F}^{\otimes n} = \begin{bmatrix} \mathcal{F}^{\otimes n-1} & 0 \\ \mathcal{F}^{\otimes n-1} & \mathcal{F}^{\otimes n-1} \end{bmatrix} \tag{2.10}$$

Where

$$\mathcal{F}^{\otimes 1} = \mathcal{F} \tag{2.11}$$

For example, $\mathcal{F}^{\otimes 2}$can be calculated in the following technique

$$\mathcal{F}^{\otimes 2} = \begin{bmatrix} \mathcal{F}^{\otimes 1} & 0 \\ \mathcal{F}^{\otimes 1} & \mathcal{F}^{\otimes 1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.12}$$

The initial format of polar codes was not systematic. That is to say; the data digits are not clearly discernible within the codeword.

Generator matrix aimed at polar codes of block N.If $v_1^N = u_1^N B_N$, then $v_{b_1,b_2,\dots,b_n} = u_{b_n,} b_{n-1}, \dots, b_1$.Polar encoder splits source word obsessed by two parts:$\mathrm{u} = (u_A, u_{A^c})$. Generator matrix for N=4 is shown in Figure (2.7).

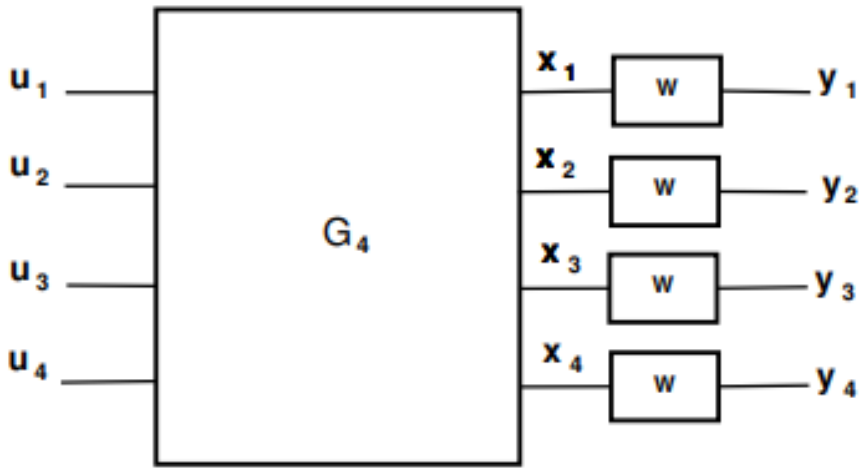


Figure (2.7): Generator matrix for N=4 [41].

2.2.2.3 Successive Cancellation Decoding (SC)

The low-complexity compilation is required for efficient code. Arikan proposed using SC to decode the polarization code. The reduced complexity of the architectures offered by SC decoding makes it suitable for area-stringent applications [42]. The SC algorithm has an inherent serial nature, so as a result of excessive delay time and a poor data transfer rate, these SC decoders are insufficient.[43]. Decoding a Polar Code can be accomplished using various strategies, including SCAN decoding [44], SCL decoding [45], BP decoding [46], and others. Here, the focus will be on introducing SC decoding. The recursion formula and butterfly algorithm are the two fundamental aspects of SC decoding. The SC method was modified to include a precomputation scheme, which resulted in the overall latency being lowered from (2n-2) to (n-1) [47,48].

2.2.2.4 SC Decoding Algorithm and Process

In successive cancellation (SC) decoder, the soft message is usually used, which is based on the log-likelihood-ration LLR form. LLRs are used in the Polar code decoder to make bit decisions during decoding. They are calculated as functions f-function which is f(L_a, L_b) or g-function which is g (u_s, L_a, L_b) where L_a and L_b are LLR values and u_s is the threshold of g-function ,this calculation depend on the decoding node position, as shown in Figure (2.8), and they get calculated in two distinct methods for polar codes depending on the position of the decoding node [49].

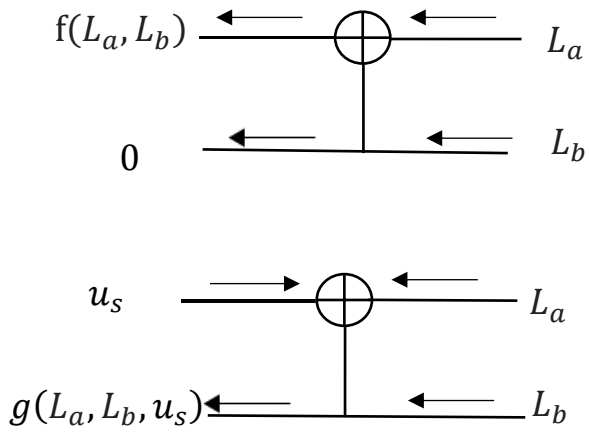


Figure (2.8): The f and g functions.

Figure (2.8) shows that the LLR values calculated with the f-function only depend on the LLR-values of nodes a and b; however, the result from the f-function also has an effect on the LLR values that were generated with the g-function. This can be observed by comparing the two sets of data.[49]

$$\mathrm{f}(L_a, L_b) = L_a \boxplus L_b = 2\tanh^{-1}\left(\tanh\left(\frac{L_a}{2}\right)\tanh\left(\frac{L_b}{2}\right)\right) \tag{2.13}$$

The *f*-function can be approximated as follows:

$$\mathrm{f}(L_a, L_b) = L_a \boxplus L_b = \ \mathrm{sign}(L_a)\mathrm{sig}(L_b)\min(|L_a|, |L_b|) \tag{2.14}$$

The g-function is calculated as follows [50]:

$$g(L_a, L_b, u_s) = (-1)^{u_s} L_a + L_b \tag{2.15}$$

Figure (2.9) shows how the f and g functions are related and connected, starting from stage 0, where the LLRs are received from the channel, until the last stage, where the LLRs are hard decoded to get the bit decisions.

At each stage j of the SC decoder, the LLR element at position i is represented by L[i][j]. It can be seen in Figure (2.9) that three stages are required for a polar code of size 8.

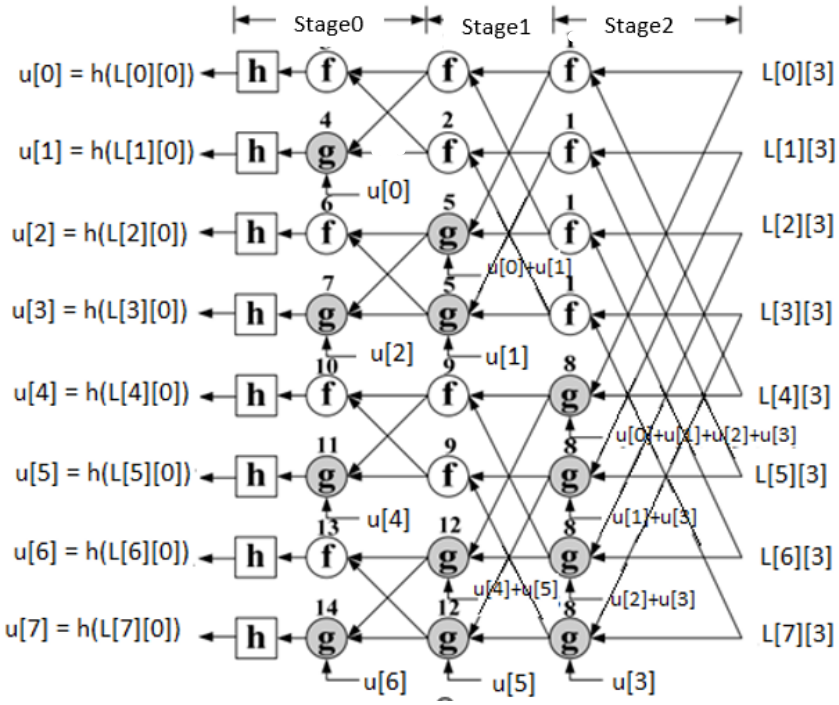


Figure (2.9): The decoding procedure of SC algorithm with *N*=8 [43].

After computing the LLR value during the final step L[i][0] with the help of the h-function, the SC decoder begins making a hard decision of the bit u[i].

$$u[i] = \begin{cases} h(L[i][0]) & hard\ decision\ (data) \\ 0 & frozen\ bit \end{cases} \tag{2.16}$$

Figure (2.10) shows the unit factor graph of the polar code. There are 8 values on the factor graph, which represent the LLR value transferred to the left and the hard bit information transferred to the right.

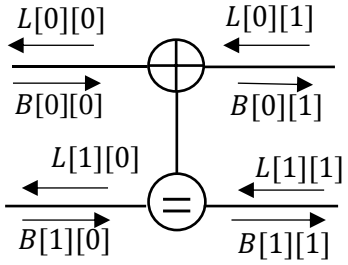


Figure (2.10): polar code decoding unit factor diagram.

$$L[1][0] = \begin{cases} L\,[1][1] + L[0][1]\,, & B[0][0] = 0 \\ L\,[1][1] - L[0][1]\,, & B[0][0] = 1 \end{cases} \tag{2.17}$$

$$B\,[0][1] = B\,[0][0] \oplus \mathrm{B}\,[1][0] \tag{2.18}$$

$$B[1][1] = B[1][0] \tag{2.19}$$

To estimate a bit in SC decoding, it is necessary to know the values of the preceding bits. Scheduling, as shown in Figure (2.11), allows the conventional SC decoder to finish the decoding operation in 2n-2 time steps, this mean that one bit is decoded every one cycle [51].

1	2	3	4	5	6	7	8	9	10	11	12	13	14
L[0][2]						B[0][2]	L[4][2]						
L[1][2]						B[1][2]	L[5][2]						
L[2][2]						B[2][2]	L[6][2]						
L[3][2]						B[3][2]	L[7][2]						
	L[0][1]		B[0][1]	L[2][1]				L[4][1]		B[4][1]	L[6][1]		
	L[1][1]		B[1][1]	L[3][1]				L[5][1]		B[5][1]	L[7][1]		
		L[0][0]	L[1][0]		L[2][0]	L[3][0]			L[4][0]	L[5][0]		L[6][0]	L[7][0]
		0	0		0	U[3]			0	U[5]		U[6]	U[7]
		frozen	frozen		frozen				frozen				

Update LLRs

Update Bits

Hard bit decision

Figure (2.11): The scheduling of SC decoder with N=8.

Step 1 of the decoding is shown in Figure (2.12): LLRs in stage 2 are calculated using the f-function and LLRs received by the channel.

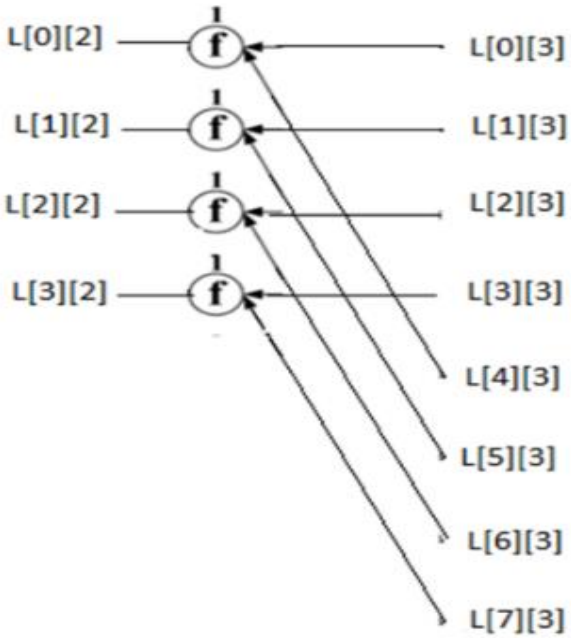


Figure (2.12): SC Decoding Step 1.

In step 2, the resulting LLRs allow the generation of 2 more LLRs in stage 1 using f-function, as shown in Figure (2.13).

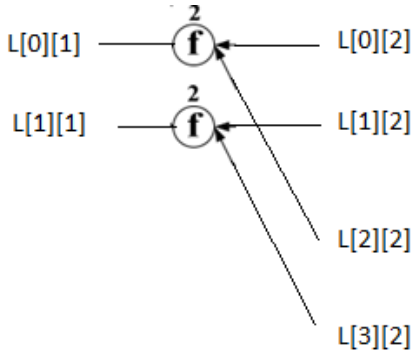


Figure (2.13): SC Decoding Step 2.

Figure (2.14): explains step 3 .in this step, the first LLR in stage 2 L [0][0] is calculated using the f-function and previous LLRs from step 2. The hard decision h-function can be used to calculate u [0], but knowing that this bit is frozen, we have u[0]=0.

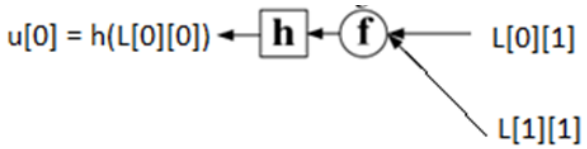


Figure (2.14): SC Decoding Step 3.

Figure (2.15) show step 4, in this step, we can use the g-function to calculate L [1][0] using previous LLRs from step 3 and the decision bit at the corresponding f-node, which is u [0]. At the same step, we can propagate the hard decision in the opposite direction to update the bits at stage 0, B [0][1] and B [1][1].

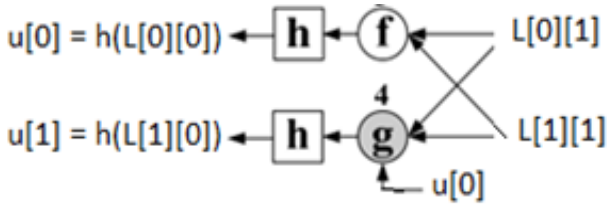


Figure (2.15): SC Decoding Step 4.

We can notice that the decoding algorithm consists of three types of operations, as shown in Figure (2.9): updating LLRs (using f and g functions), hard decisions (using the h function) and updating bits (using the XOR function) [52]. The algorithm continues in a similar way to obtain all the bit decisions**.**

2.3 Field Programmable Gate Array (FPGA)

FPGA is a semiconductor device which can program by the designer and reconfigure at any time, and the process of data at a higher speed than digital signal processor (DSP) and microprocessors, where the implementation in parallel structure, these features make FPGA device suitable for the implementation of error correction codes such as polar code.

FPGA comprise a series of logic cells linked by a programmable metal interconnect and an array of I/O cells, as shown in Figure (2.16) [53].

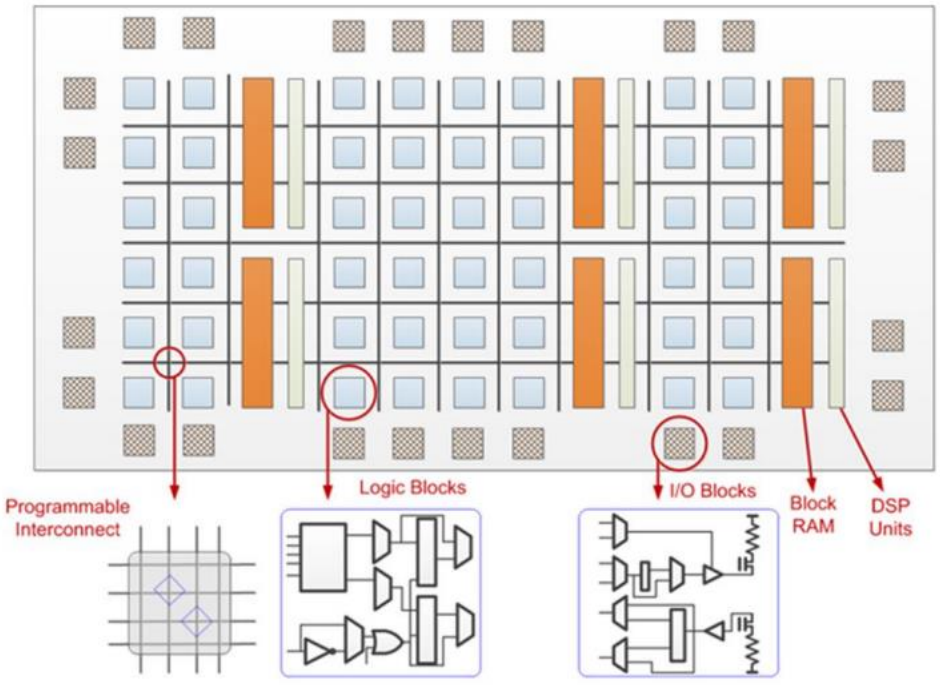


Figure (2.16): Basic block diagram of an FPGA.

The cells and programmable interconnect switches are built as static RAM (SRAM), so the current configuration is stored in these RAM elements. To reconfigure, the memory elements must simply be loaded with new values; thus, the hardware can be changed entirely in milliseconds [54].

The three basic blocks comprising the FPGAs are as follow [55].

a. The Configurable logic blocks (CLBs), where the functions are worked.

b. Input/Output blocks (IOBs), these blocks make the FPGA connects with the other elements of the application.

c. Interconnect, it is used to write between CLB and from IOBs to CLBs. In any digital design, there are three necessary physical terms. These concepts in an FPGA reflect processing data as; Throughput refers to the quantity of information transmitted per clock cycle.

The time between data input and processed data output refers to latency. Timing refers to the delays between sequential elements in logic [56].

The FPGA design flow introduces in Figure (2.17) [57]

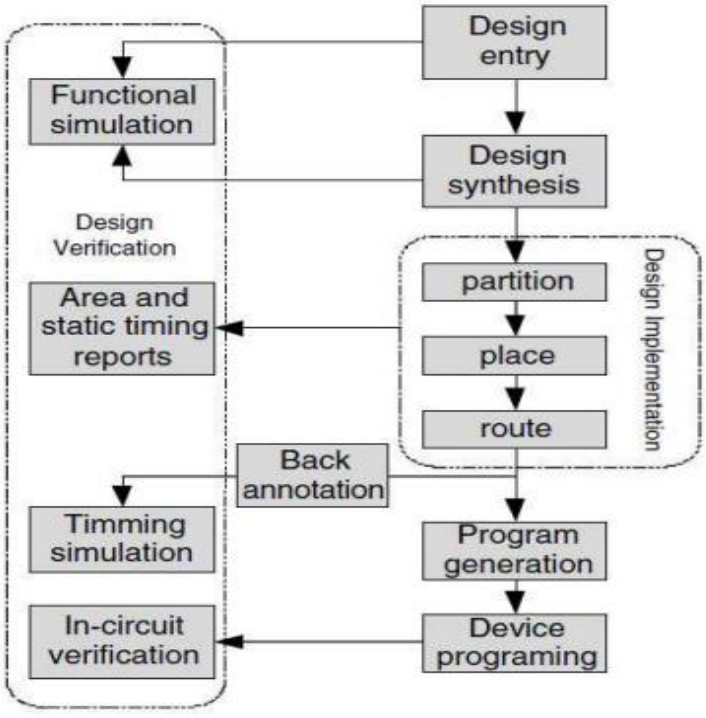


Figure (2.17): FPGA design flow.

The consecutive process blocks are described as follows [58]:

i. Design Entry: Schematic editor or HDL (Verilog and VHDL) are used to make the design file.
ii. Design Synthesis: An automatic process to convert a high level of logic abstraction (HDL) to a lower level of logic abstraction.
iii. Partition (Mapping): A specific physical element is assigned for each logic element that rightly implements the logic function in a configurable device
iv. Place: A process to specify locations in the FPGA chip for map logic.
v. Route: A process of a link of the mapped logic.
vi. Program Generation: A bit-stream file is generated to program the device.
vii. Device Programming: The bitstream is downloading on the FPGA.
viii. Design Verification: The simulation can be done to check functionalities. FPGAs have many advantages and disadvantages, so they are used in many applications, as shown below [57,59].

2.3.1 Utilization Cases for FPGAs

FPGAs' parallel processing architecture, which allows them to address complicated computation issues, and their re-programmability make them ideal for modern computing applications. Examples of where FPGAs' great adaptability has been put to use are:

- **Medical Gear.** Video processing using FPGAs is becoming more commonplace, and this includes the highly specialized field of Optical Coherence Tomography, in which an FPGA is used to construct an image from an optical sensor placed inside a blood vessel. FPGAs are used to improve the

quality of images captured by x-ray, ultrasound, and other imaging instruments.

- **Automotive.** Light Detection and Ranging LiDAR uses FPGAs in cars to piece together images from data provided by the laser beam, one pixel at a time. Video is created from the photos in real time. In autonomous driving, they are used to rapidly evaluate video for the presence of obstacles or the road's edge. They process high-speed communications within the car, are employed in security systems to encrypt or decrypt messages fast, and increase efficiency and energy usage.

- **Commerce and manufacturing.** FPGA chips are a game-changer for automation and security, allowing businesses and factories to better protect themselves from online attacks while also facilitating more significant levels of automation in the workplace. Automated stock trading they are also utilized to evaluate stock transactions in milliseconds.

- **Communication systems.** In order to improve network capacity, coverage, and overall quality of service while decreasing delays and latency, especially with data processing, FPGAs are widely utilized in communication systems. FPGA is widely used in enterprise server and cloud applications. Hardware-programmable gate arrays (FPGAs) are also utilized in SDRs. Instead of employing a modulator, the RF waveform is created in the FPGA itself.

- **Defense Department.** FPGAs are also used by the military to increase the safety of weapons and vehicles and to automate routine tasks.

2.3.2 Differentiating FPGA and Microprocessor:

Field Programmable Gate Arrays (FPGAs) are programmable logic circuits that may be programmed electrically to execute a wide variety of tasks. When compared to a microprocessor or an ASIC, how does an FPGA fare?

One may compare a microprocessor to a stripped-down Central Processing Unit. A set of instructions included in a program are carried out. The primary distinction between FPGAs and microprocessors which are shown in figure (2.18) and (2.19) respectively lies in the level of complexity required to program each device. Even though the complexity of both can rise, microprocessors are often more complex than FPGAs [60]. This occurs as a result of the system's pre-built procedures. Microprocessors contain a predetermined set of instructions that must be learned by programmers in order to produce a functional application. The microprocessor already has a dedicated block programmed for each of these instructions. Since an FPGA's main selling point is its ability to be programmed in the field, it lacks any hardwired logic blocks.

Figure (2.18): FPGA.

Each node in an FPGA is a switch that the user can enable or disable. How the logic of each block is determined depends on this. Some people compare learning to program an FPGA with learning assembly language since HDL, or the Hardware Description Language is a low-level language. As semiconductors and electronics have improved and become cheaper, FPGAs and microprocessors have increasingly merged into a single component. The integrated package benefits greatly from this increased adaptability. The majority of the processing is handled by the microprocessor, while the more specialized jobs are delegated to an FPGA block [61]. In this way, you can enjoy the advantages of both options. The microprocessor can deal with the routine duties, and the custom FPGA blocks will allow you to include the special ones. Contrast this with an FPGA, which is much simpler. In

contrast to microprocessors, FPGAs may execute arbitrary code. Combinations of FPGAs with microprocessors are common.

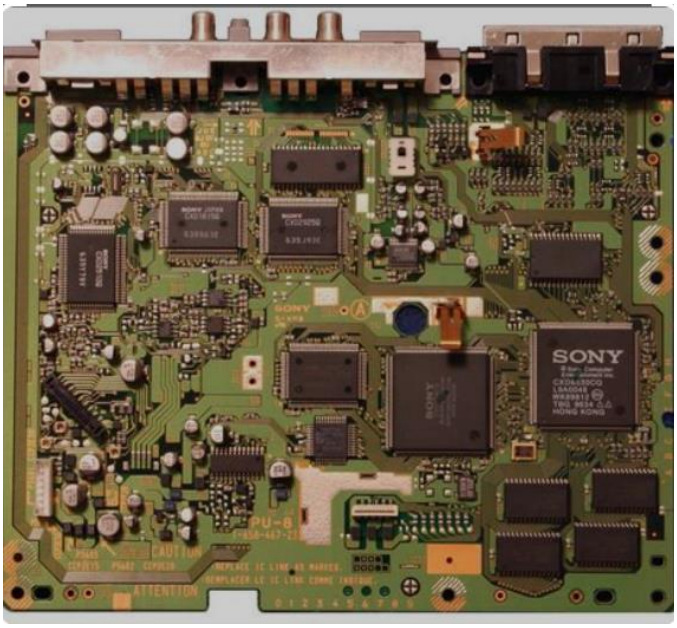


Figure (2.19): Microprocessor.

2.4 High-Level Synthesis (HLS) vs (VHDL)

In recent years, more than thirty tools [62] have been developed for use in the fast-developing field of High-Level Synthesis (HLS) within Electronic Design Automation (EDA). This shift is largely attributable to the growing demand for a fast, in addition, a reliable design platform that, given a high-level model defined in a language like C, C++, or a C-based variant like System C, generates a low-level counterpart that can be used to create hardware using technologies like FPGAs and ASICs [63]. Working at the Register Transfer Level (RTL) can be difficult, but this technology can help [64]. Due to the scarcity of VHDL/Verilog professionals, even non-specialist programmers need to acquire a high level of specialization to facilitate effective model building.

On the other hand, VHDL also Verilog aren't as robust as high-level languages, which results in highly lengthy source codes, which in turn raises the likelihood of coding errors and slows down the process of refining the design for improvements. With their ability to rapidly generate production-ready RTL models from higher-level models, HLS tools have become increasingly popular. Vivado HLS, which comes as a component of the Vivado Design Suite, is a top-tier HLS tool [65]. This is shown in Figure (2.20)

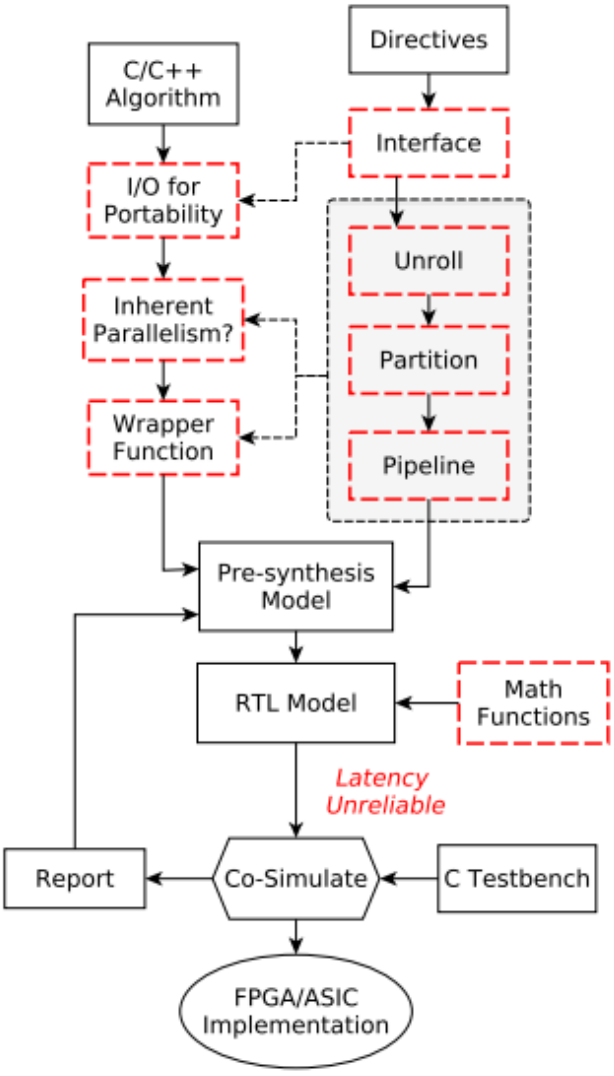


Figure (2.20): High-Level Synthesis Flow – Enhanced.

C/C++ is used to code the input models, and either cycle-accurate or untimed VHDL/Verilog RTL models (including System C) are produced at the end. In addition, it provides automatic testbench development, which makes co-simulation an accessible option for verifying behavioural and functional correctness. There are, however, considerations that must be made whenever Vivado HLS is being used.

Figure (2.21) shows the sequence of operations in the HLS program that ends with the conversion of code to IP which can be utilized by other platforms such as Vivado Design Suite, System Generator, and Xilinx Platform Studio.

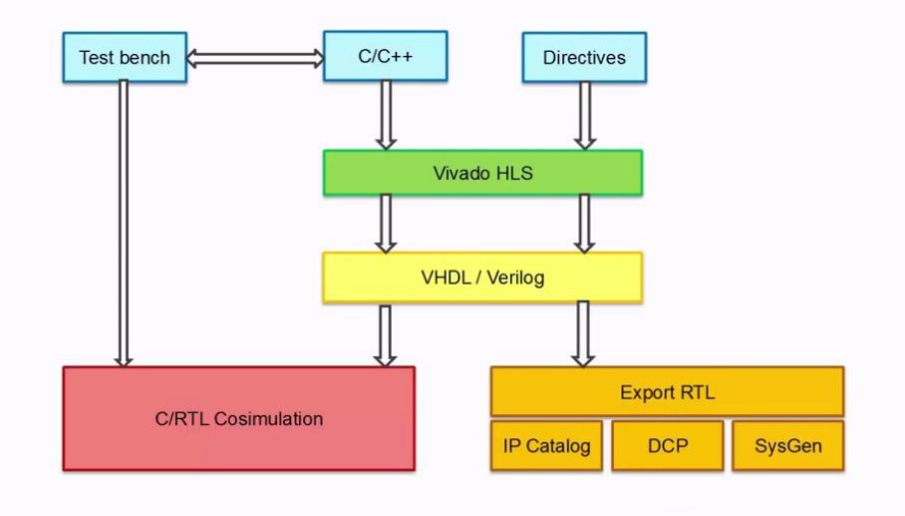


Figure (2.21): Vivado HLS Design Flow.

Different parts that comprise the coded communications system such as pseudorandom binary source, polar encoder, AWGN channel, and SC decoder are completely implemented as IPs using the HLS platform, and they are discussed in the next sections in details.

After ensuring that the code works properly by programming it in C++ language and running C simulation in HLS, different improvements are applied to the HLS code such as (dataflow, pipeline, unrolling loops, array partition, etc.) [66].

Chapter Three

The Proposed system

3.1 Introduction

Polar codes, which are gaining popularity as one of the most advantageous capabilities for creating error correction codes (ECC) due to the simplicity and low complexity of their encoding and decoding, have attracted a lot of attention in latest years; they used in 5G, new radio (NR). The proposed work includes the software and hardware design and implementation of polar code using successive cancellation decoder. This work is composed of two distinct sections: The first is the simulation of the designed project with the help of the VIVADO HLS program. The proposed system was built to present the performance of polar code in terms of BER, FER, frame length, number of simulated frames, coding rate, generator polynomial, and channel condition (SNR). The second is Hardware implementation of polar code based on FPGA device using Digilent Genesys2 board based on kintex7 (xc7k325tffg900-2 as a core).

3.2 System Model

In this work, the binary information sequence is generated at the transmitter with uniform distribution and is used as the information that should be transmitted successfully to the destination point. The information bits entering the polar encoder to generate the codewords. These codewords are transmitted through the AWGN channel and affected by the noise of this channel. At the receiver sides SC decoder algorithm is applied to obtain the original information. The binary source,

encoder, channel, and decoder are programed in C++ language using the VIVADO HLS platform.

The following points can summarize the main procedures that are followed in this study to simulate and implement the polar coded systems:

1- Initially, to design the parts of the tested system use C++ language with the aid of the Vivado High-Level Synthesis (HLS) software. The Vivado HLS is a tool to transform the design into a Register Transfer Level (RTL) and export it to an RTL package in the desired IP (intellectual property) format.
2- The HLS offers the ability to test the functionality of the generated IP individually, using a test bench program. All IPs are successfully tested individually before they are assembled in the Vivado IP integrator.
3- Utilization of resources is a key consideration (FF, Look-Up Tables, Block RAM, and DSP slices) and latency reported by Vivado HLS when creating hardware design. These reports are crucial to the designer to accommodate the needed resources and to not exceed that offers by the FPGA device.
4- The produced IPs are assembled and linked together at the Vivado IP integrator. Microblaze, Memory Interface Generator "MIG," processor system reset, Microblaze Debug Module "MDM," AXI interconnect, clocking wizard, AXI timer, and AXI UARTlite are some examples of the essential IP blocks that must be provided by Xilinx or one of its other partners in order to guarantee that the system is functioning appropriately.
5- The FPGA is configured using the JTAG-USB bridge port by the bitstream file. Vivado was the program that was employed in the generation of the bitstream. The bitstream from Vivado was exported to Xilinx Software Development Kit (XSDK), which is an integrated design tool that offers a

comprehensive environment for the development of software applications that are aimed at Microblaze.

6- The Microblaze is a soft-core processor and is responsible for controlling the process of executing the IPs via the AXI bus using their drivers that, in advance, are provided by the Vivado platform.

7- The results like BER, FER, the average time of constituent SC and the whole polar decoder, and average number of iterations can be displayed using the USB-UART port that is provided by the AXI UART lite IP and displayed on the SDK console or a terminal program like Tera Term software.

3.3 Simulation of Polar Code Using VIVADO HLS

The utilization of High-Level Synthesis (HLS) tools is one technique for creating on FPGA that is always being improved upon. HLS provides a design flow that allows algorithms to be implemented utilizing high-level languages like as C or C++, in addition to system C. The HLS tools will interpret these implementations, and then they will build Register Transfer Level (RTL) models of the algorithm in a hardware specification language such as VHDL or Verilog. For efficient design in Vivado HLS, it is important to understand the many implementation options, such as directives, data types, and interfaces, as well as the circumstances under which each option may be utilized most effectively. It also implies having the ability to understand and read HLS reports and to evaluate the output using the many analytic tools that are made available. There are two steps to verify the design:

- C Validation: Validating the algorithm is correct before synthesis usually by using the test bench with golden data.

- RTL Verification: Vivado HLS can co-simulate the RTL with the original test bench.

3.3.1 Binary Source IP

this IP is to generate a pseudo-random binary source with uniform distribution. It also stimulates the physical information that might be the output of the source encoder. The pseudorandom function is based on LFSR (Linear Feedback Shift Register) technique. An LFSR is a type of shift register in which the input bit is controlled by the XOR operation performed on some of the bits that make up the value of the shift register as a whole. Because the functioning of the register is deterministic, the stream of values produced by the register is fully governed by its current state (or previous state). On the other hand, an LFSR that has its feedback function carefully tuned can generate a string of bits that has both a cycle and the appearance of being random.

A C-simulation test is carried out to generate binary samples that reveal an approximately equal probability of 0 and 1. The linear feedback shift register (LFSR) is shown in figure (3.1) with generator polynomial $g(x) = x^{32} + x^{22} + x^2 + x + 1$.

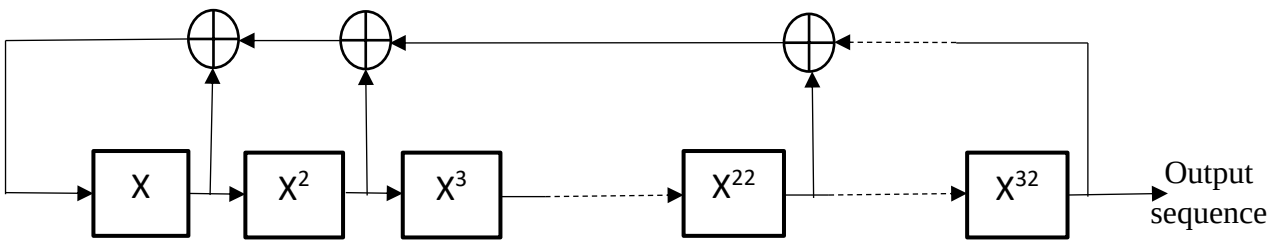


Figure (3.1) linear feedback shift register (LFSR).

Figure (3.2) shows the generated Binary Source IP, the different interface buses like AXI lite, AXI master, and some single wire control buses like ap_clk, ap_rst_n, and interrupt are used to connect the IP with the MicroBlaze and other IP cores.

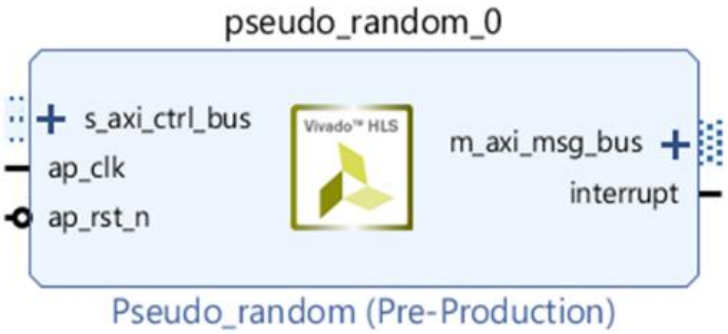


Figure (3.2): Binary source IP core.

The code for the C++ program used to implement this IP is represented by flow chart as shown in figure (3.3). This flow chart represents the output of LFSR.

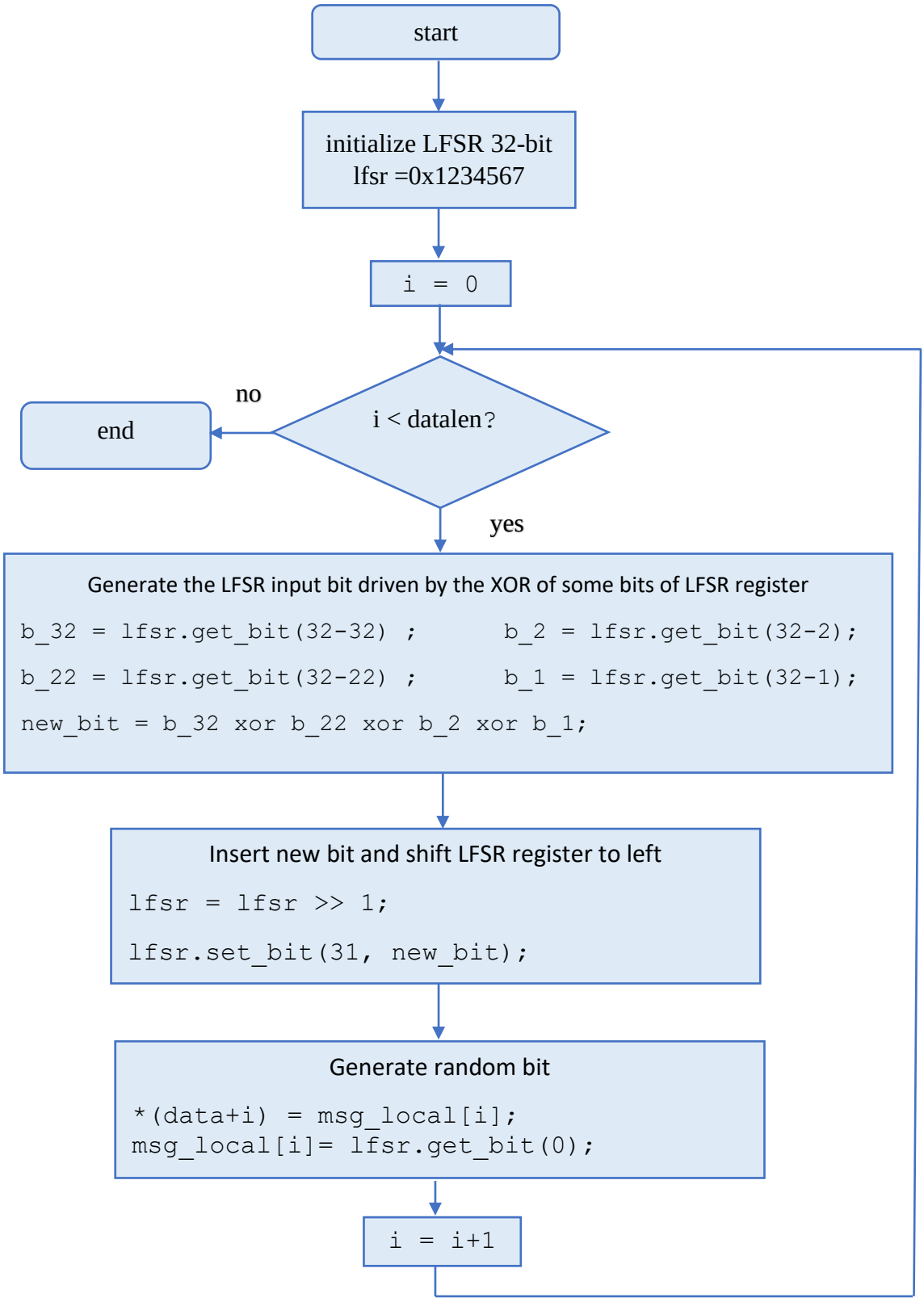


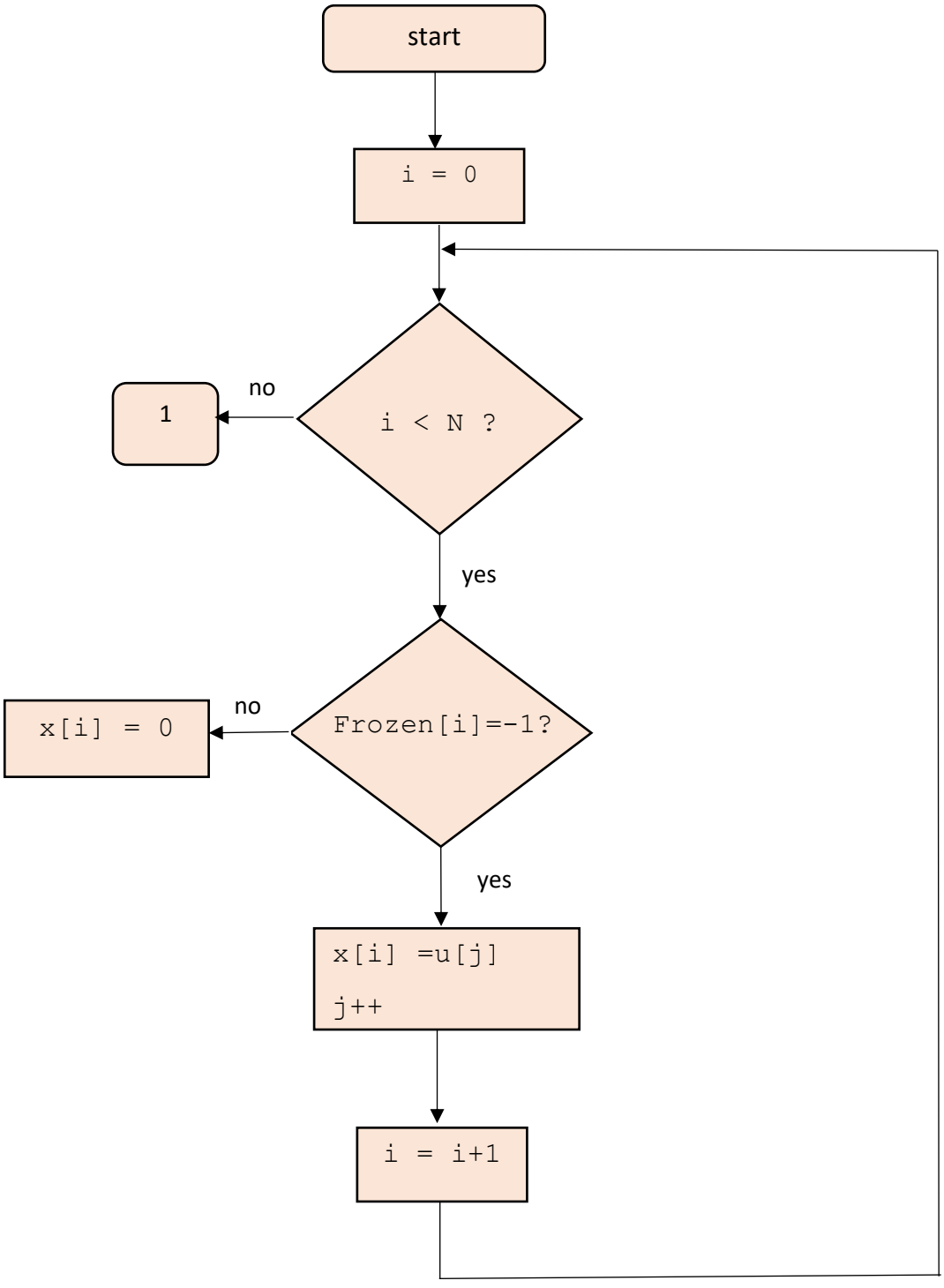
Figure (3.3): Binary Source IP flow chart.

3.3.2 Polar Code Encoder

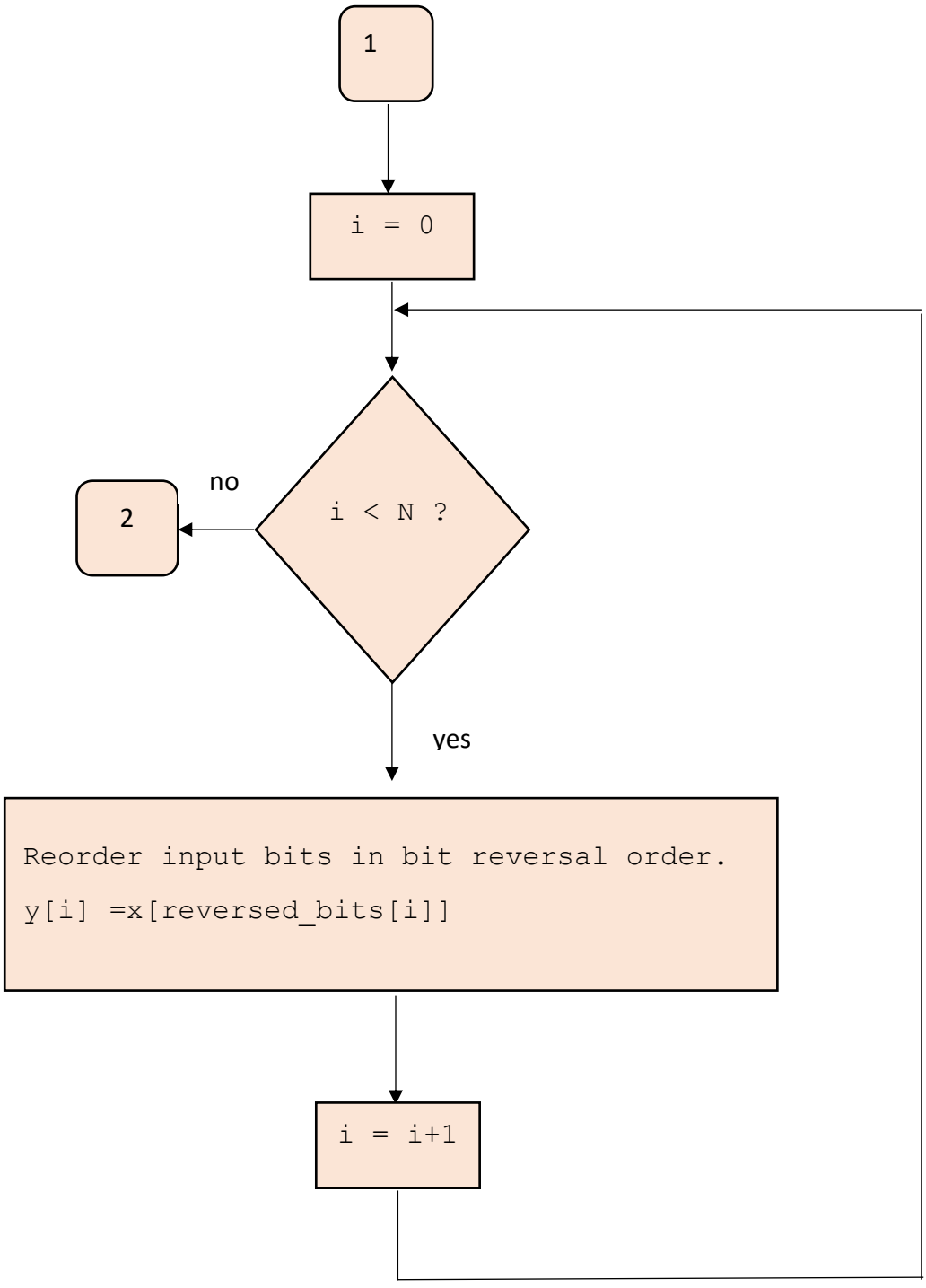
Encoders are devices whose purpose is to take information from a sender and convert it into a code word that can then be sent to a receiver. Since there is a possibility that the signal is corrupted during transmission due to noise, the encoder has the additional responsibility of adding security to the message in such a way that the decoder will be able to reconstruct the message even though some bits might be flipped.

The goal of Polar encoding makes N copies that are combined and split into different virtual channels. As explained in Chapter two the virtual channels are either" noiseless" or completely" noisy" channels. The noise which is referred to is the information which is added from other channels. The more channels that are polarized the larger the gap between the noiseless and noisy channels becomes. The combination of virtual channels is done by using a Kronecker product of the equation (2.10). where the rows of the matrix represents the virtual channels a bit should be transmitted on and the columns represent whether or not the bit should be XOR:ed when combining the virtual channels. The flow chart of the C++ program s code is given in the figure (3.4).

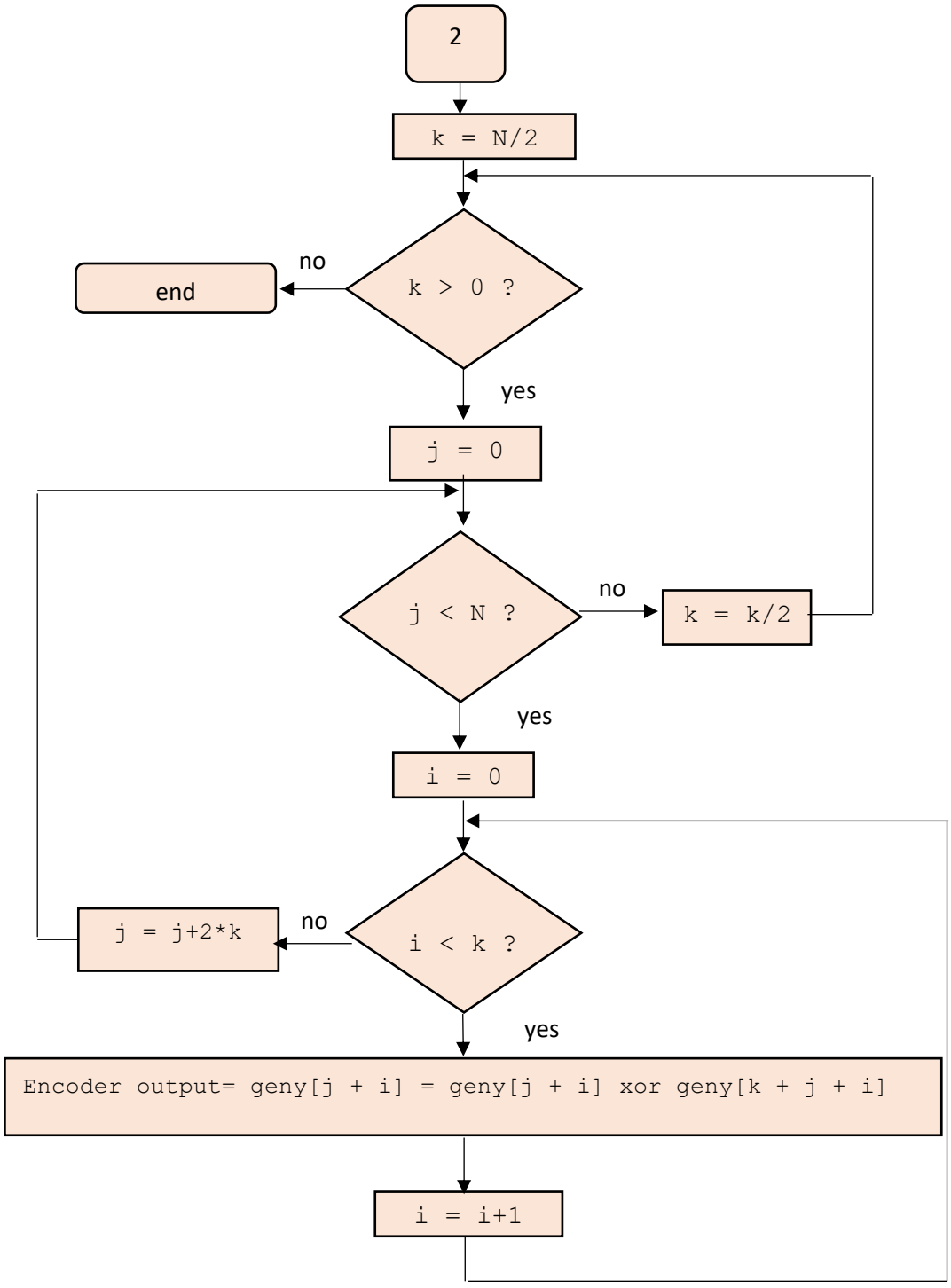
The first step in the encoder is to fix the known frozen bits to 0. The second step is reordering the input bits in the bit reversal order. The third step consists of the recursive building of the butterfly structure through 3 nested loops, where k represents the stage index and i and j are the indexes of each butterfly structure within the same stage.



(a)


1
i = 0
i < N ?
no
2
yes
Reorder input bits in bit reversal order.
y[i] =x[reversed_bits[i]]
i = i+1

(b)



(c)

Figure (3.4): (a) first step, (b)second step and (c) third step of polar encoder IP flow chart.

The IP that is generated by HLS and performs polar encoding is presented in figure (3.5).

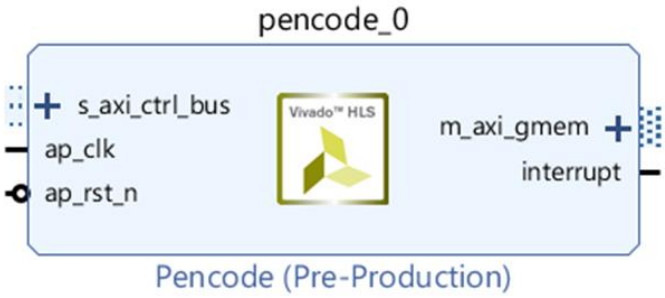


Figure. (3.5): polar encoder IP core.

3.3.3 AWGN Channel IP

IP This channel is a well model for many deep space communication and satellite links, while it is not good for almost earthly links since the existence of terrain blocking, the multipath, the interference, and many other effects. AWGN for the path of the terrestrial model is used commonly to indicate the background noise of the channel that studied. The interference, multipath, ground clutter, terrain blocking, also the self-interference are the effects that the modern radio systems faced in terrestrial operation [67, 68]. The Gaussian noise samples are generated using Box-Muller Algorithm [67]. v_1, v_2 are independent samples chosen from the uniform distribution. The generated Gaussian noise samples have zero mean, unity-variance, and the probability density function is given by [67]

$$p(n) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{n^2}{2\sigma^2}\right) \qquad (3.1)$$

where σ^2 is the variance of the noise signal n which is defined by;

$$\sigma^2 = \left(2R_c \frac{E_b}{N_o}\right)^{-1} \qquad (3.2)$$

The Flowchart of the Gaussian noise generator is shown in figure (3.6)

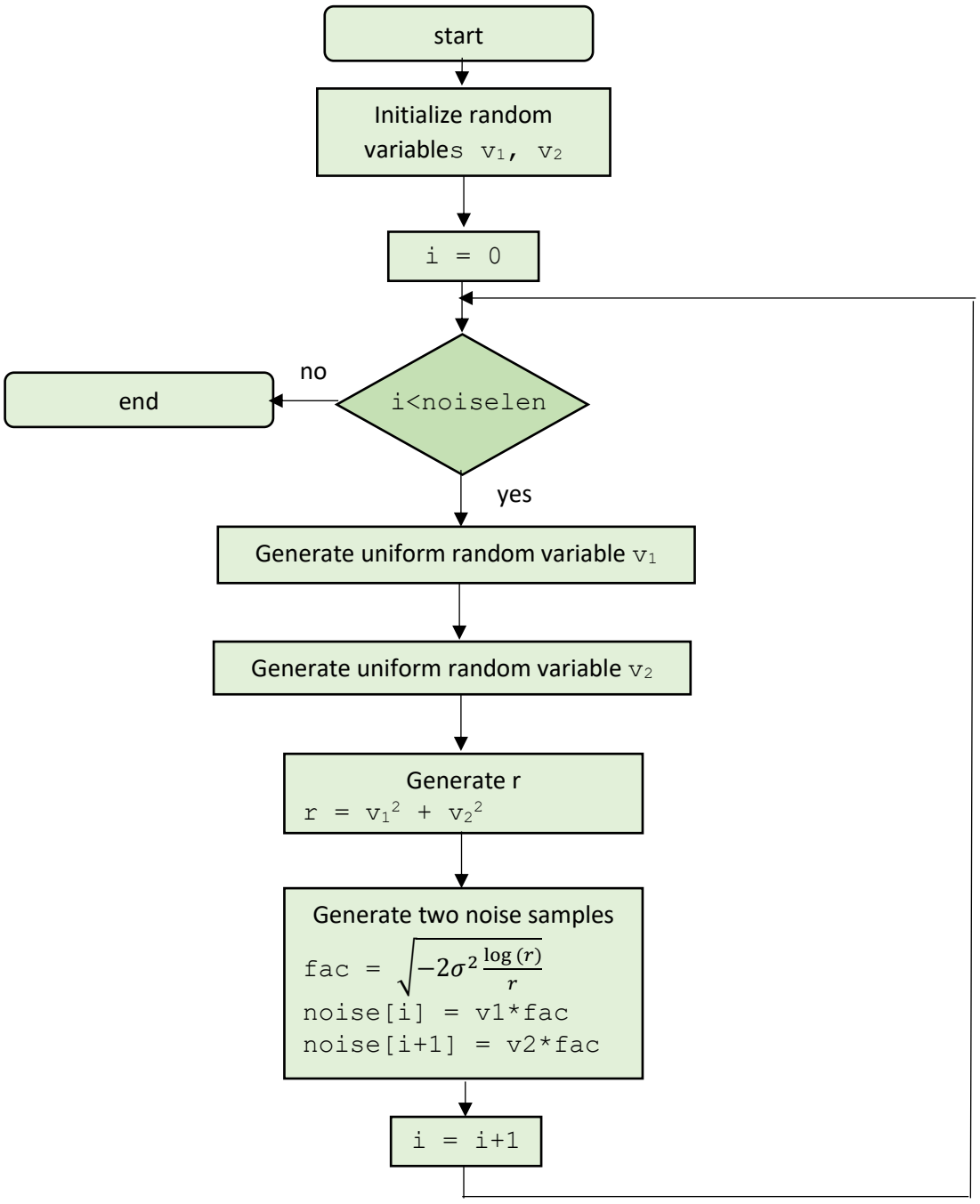


Figure (3.6): AWGN channel IP flow chart.

Figure (3.7) shows the AWGN channel IP.

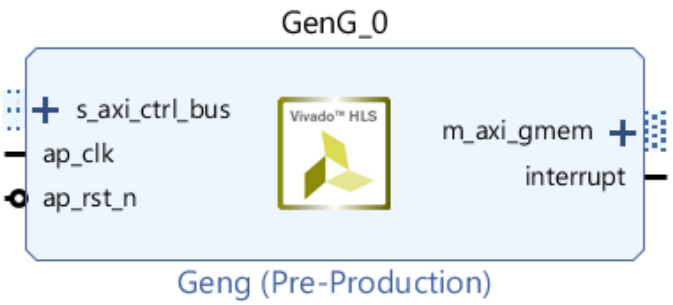


figure (3.7): Gaussian source IP core.

3.3.4 Polar Code Decoder

Polar codes were introduced with a successive cancellation (SC) decoding algorithm. Successive cancellation implies that the bits are decoded one at the time in a specified order. In general, the decision of any bit is influenced by all previous decisions. SC decoder is used in the system which explained in section (2.2.2.3).

The flow chart that utilized in the design of the SC decoder is given by Figure (3.9) and the generated IP is shown in figure (3.8).

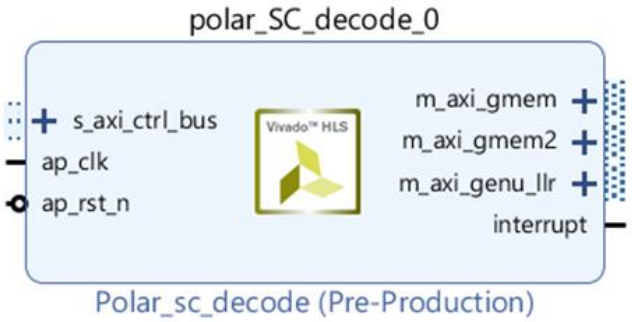
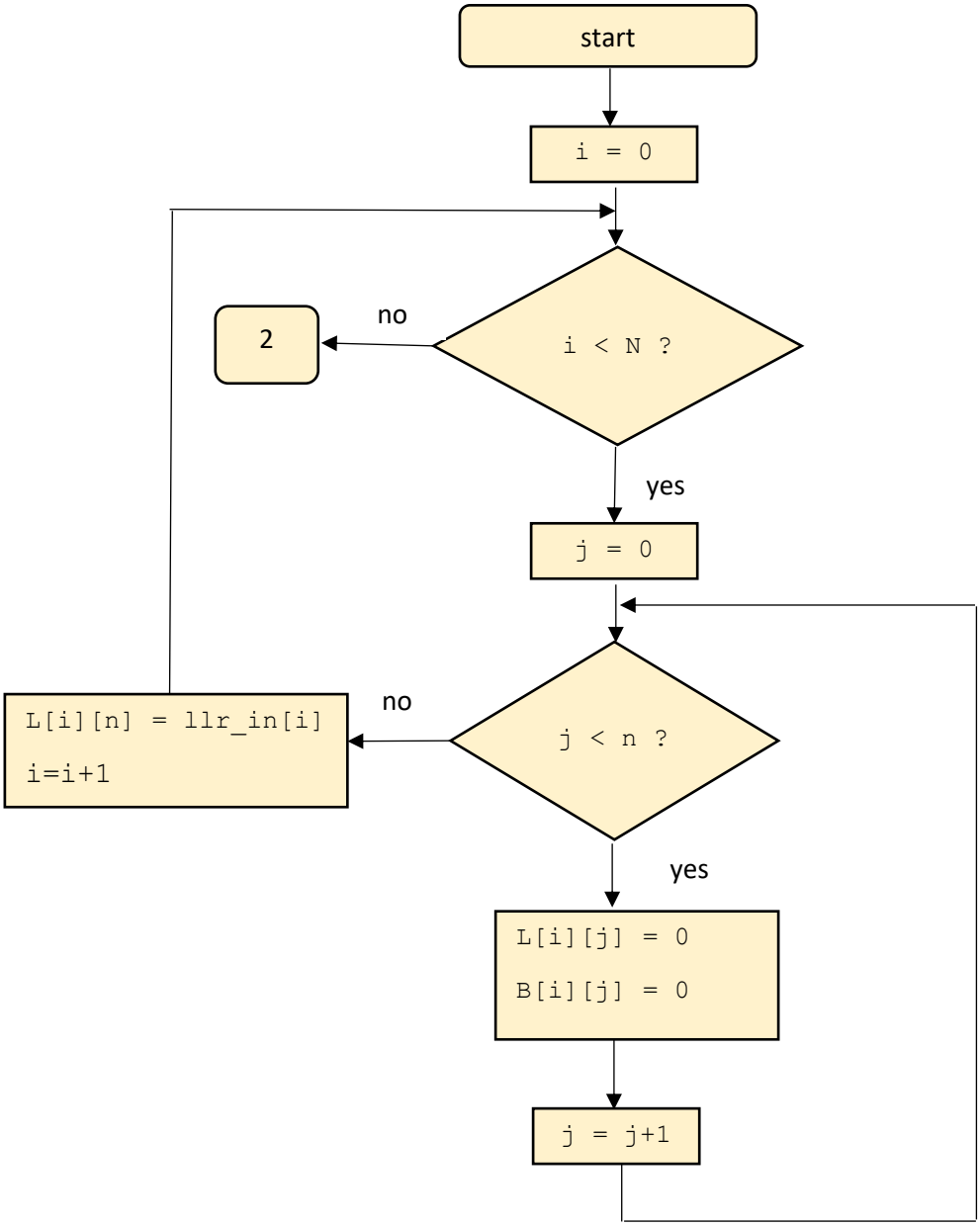


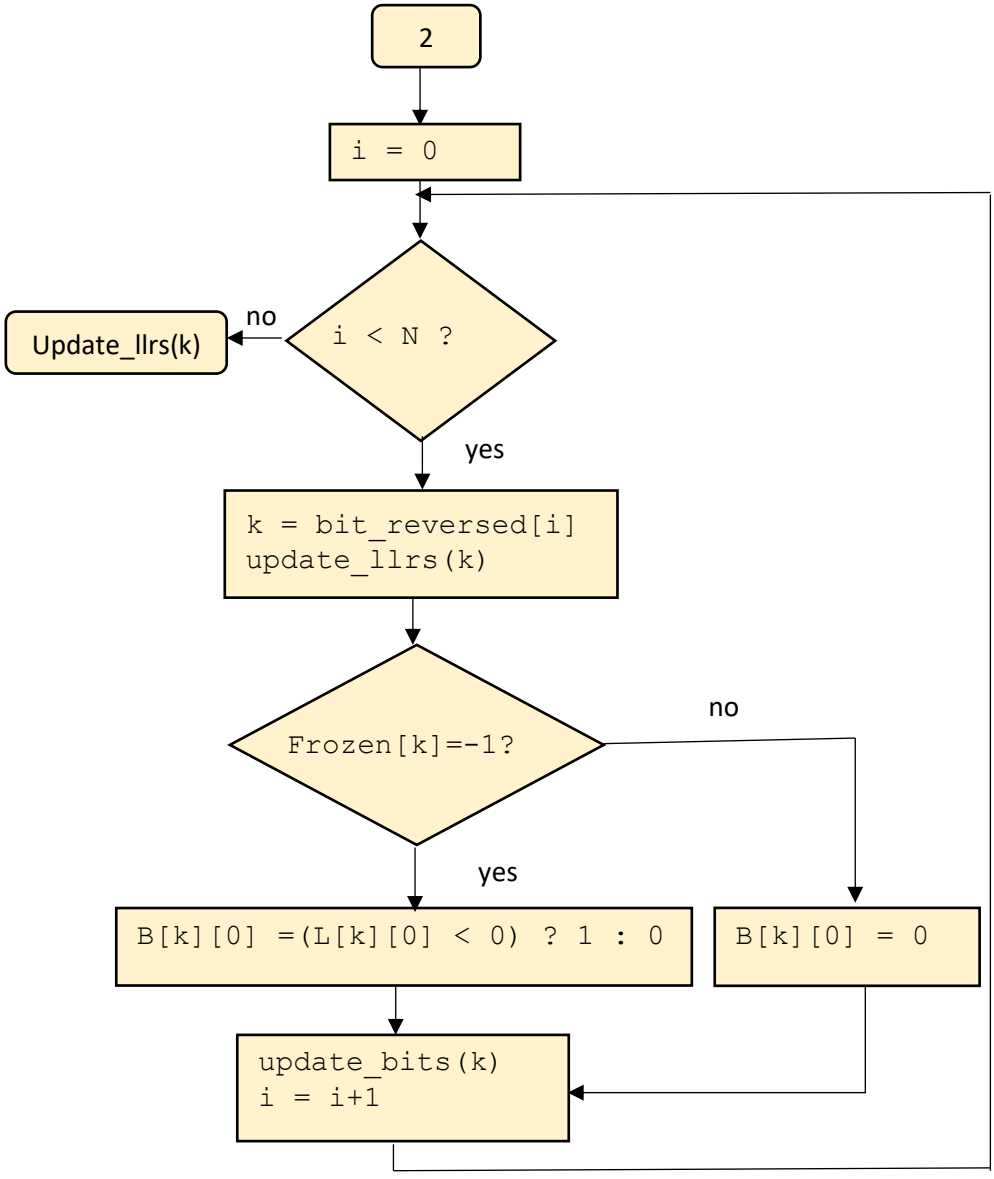
figure (3.8): SC decoder IP core.

The first part of the flow chart is shown in figure (3.9. a). It consists of creating the LLRs and bits arrays and reading the input LLR in array considered as the first stage of LLRs.



(a)

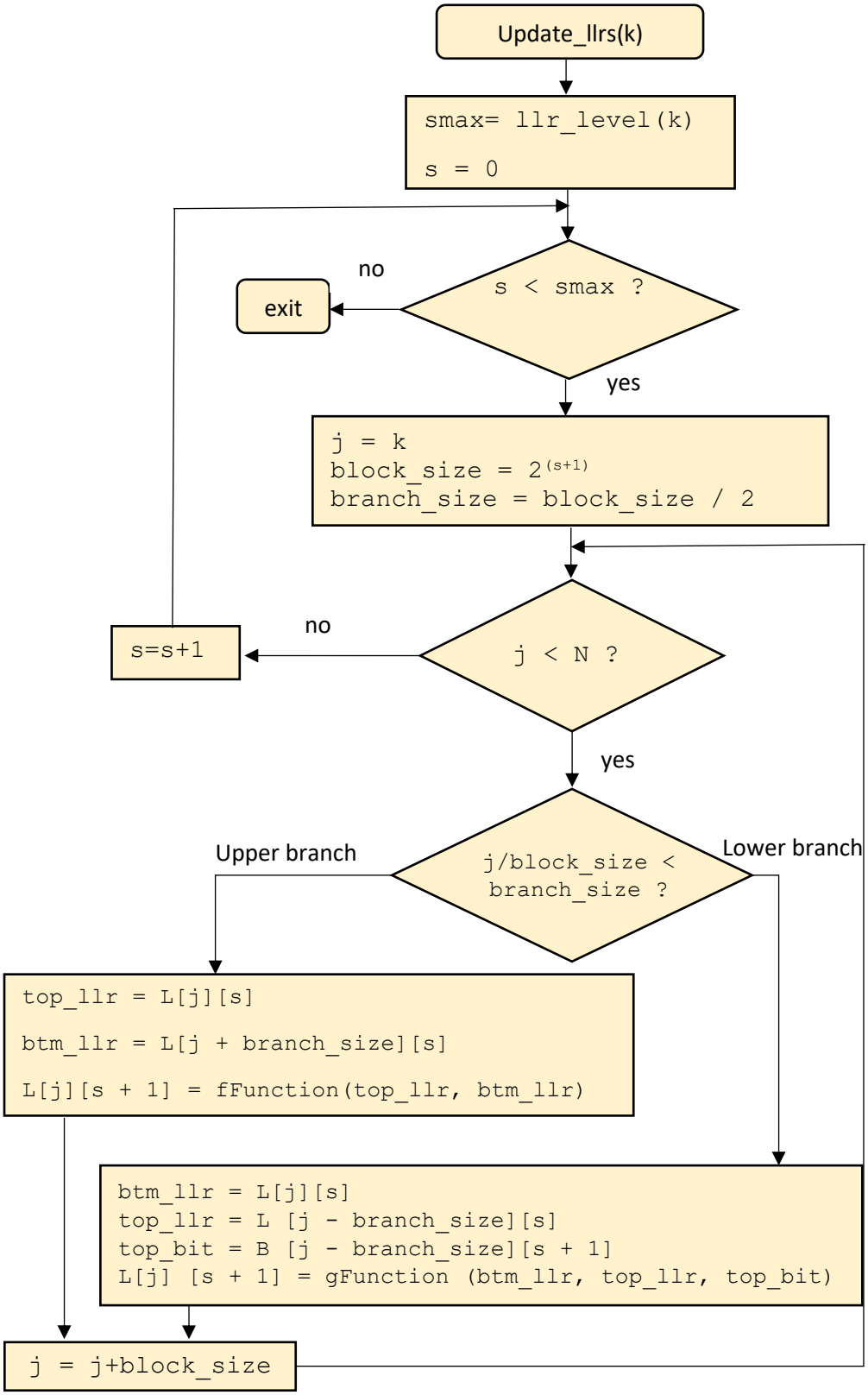
The second part of the flow chart is the main part. In this part we make a loop over all bits, see figure (3.9.b). In each loop of index k, we first update the corresponding LLRs, then we make hard decision at output k, and last, we propagate (in the opposite direction) the hard decision just made to update the bits in the corresponding stages.



(b)

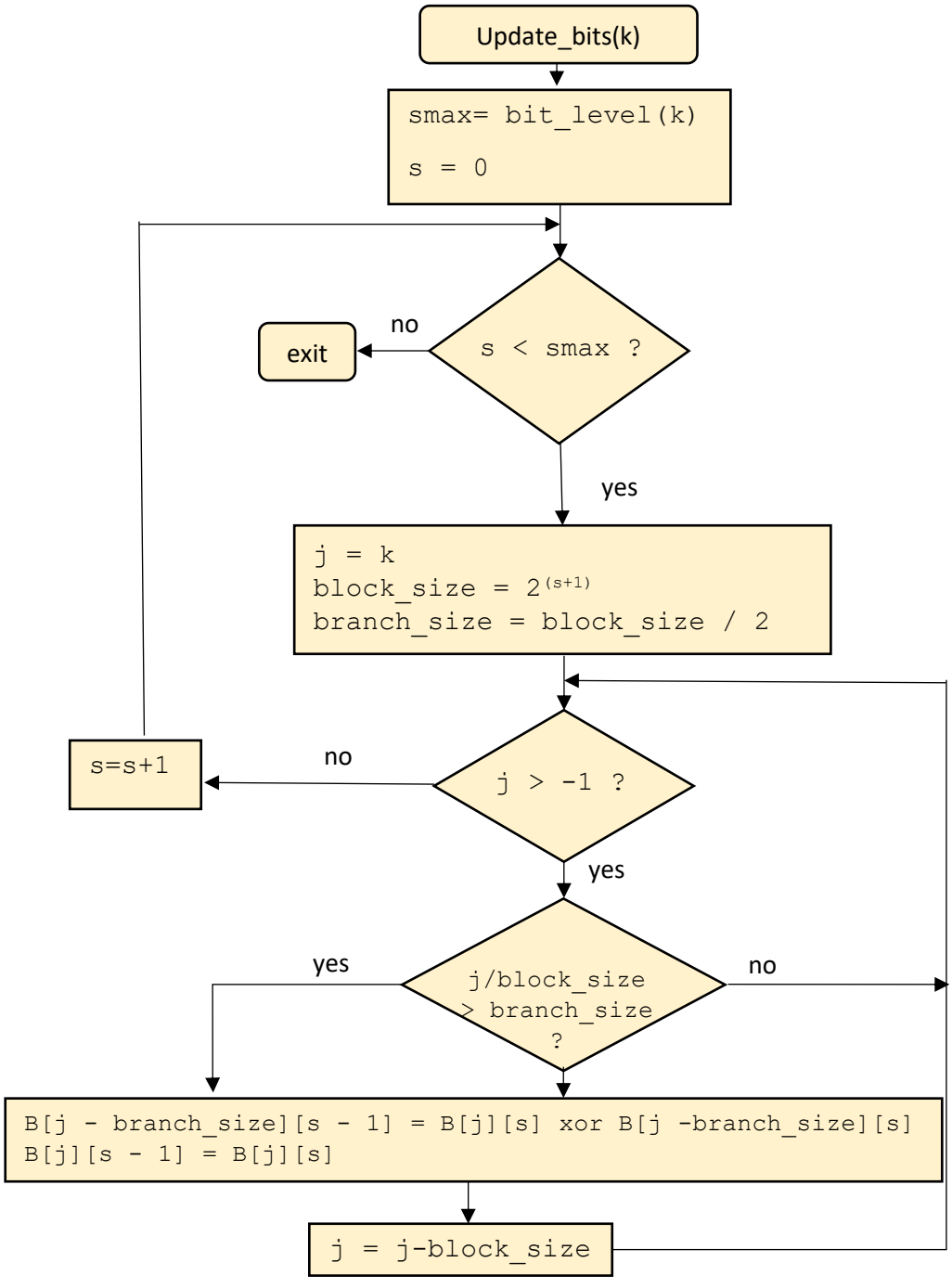
The third part of the flowchart is updating LLRs as shown in figure (3.9.c).

When going through the stages, the positions of the nodes are calculated (upper branch and lower branch) to apply the proper function (f or g)


Update_llrs(k)
smax= llr_level(k)
s = 0
s < smax ?
no
exit
yes
j = k
block_size = 2^(s+1)
branch_size = block_size / 2
j < N ?
no
s=s+1
yes
j/block_size < branch_size ?
Upper branch
Lower branch
top_llr = L[j][s]
btm_llr = L[j + branch_size][s]
L[j][s + 1] = fFunction(top_llr, btm_llr)
btm_llr = L[j][s]
top_llr = L [j - branch_size][s]
top_bit = B [j - branch_size][s + 1]
L[j] [s + 1] = gFunction (btm_llr, top_llr, top_bit)
j = j+block_size

(c)

The flowchart of updating bits is shown in figure (3.9.d). When going through the stages in the opposite direction, the positions of the nodes are calculated to update the bits using the butterfly structure.



(d)

Figure (3.9): polar decoder IP flow chart.

3.4 Assembling the Proposed System Using Vivado IP Integrator

After confirming the correct implementation of all units under test (binary source, polar encoder, AWGN channel, and SC decoder) individually using test bench programs, they converted to IP cores. These cores comprise the main parts of the implemented communication system. Figure (3.10) shows the steps that are followed to generate the bitstream file that is used to program the FPGA.

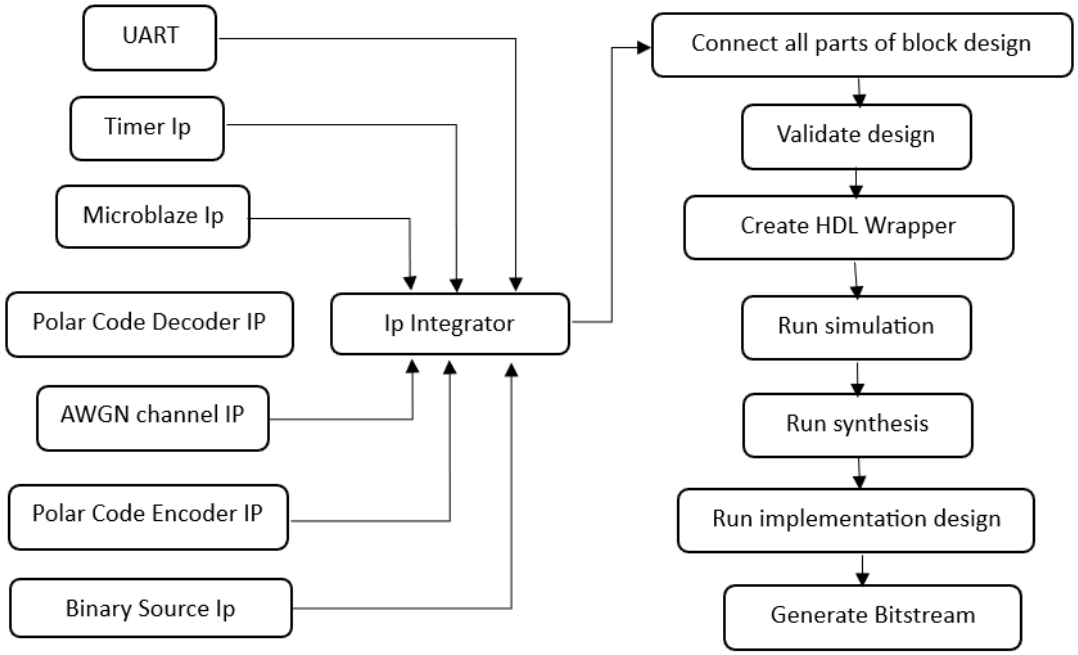


Figure (3.10): Sequence of operations within Xilinx Vivado.

Figure (3.11) shows the previously mentioned IPs and other necessary cores like the Microblaze soft processor and memory units which are all assembled in the IP integrator as a Vivado project.

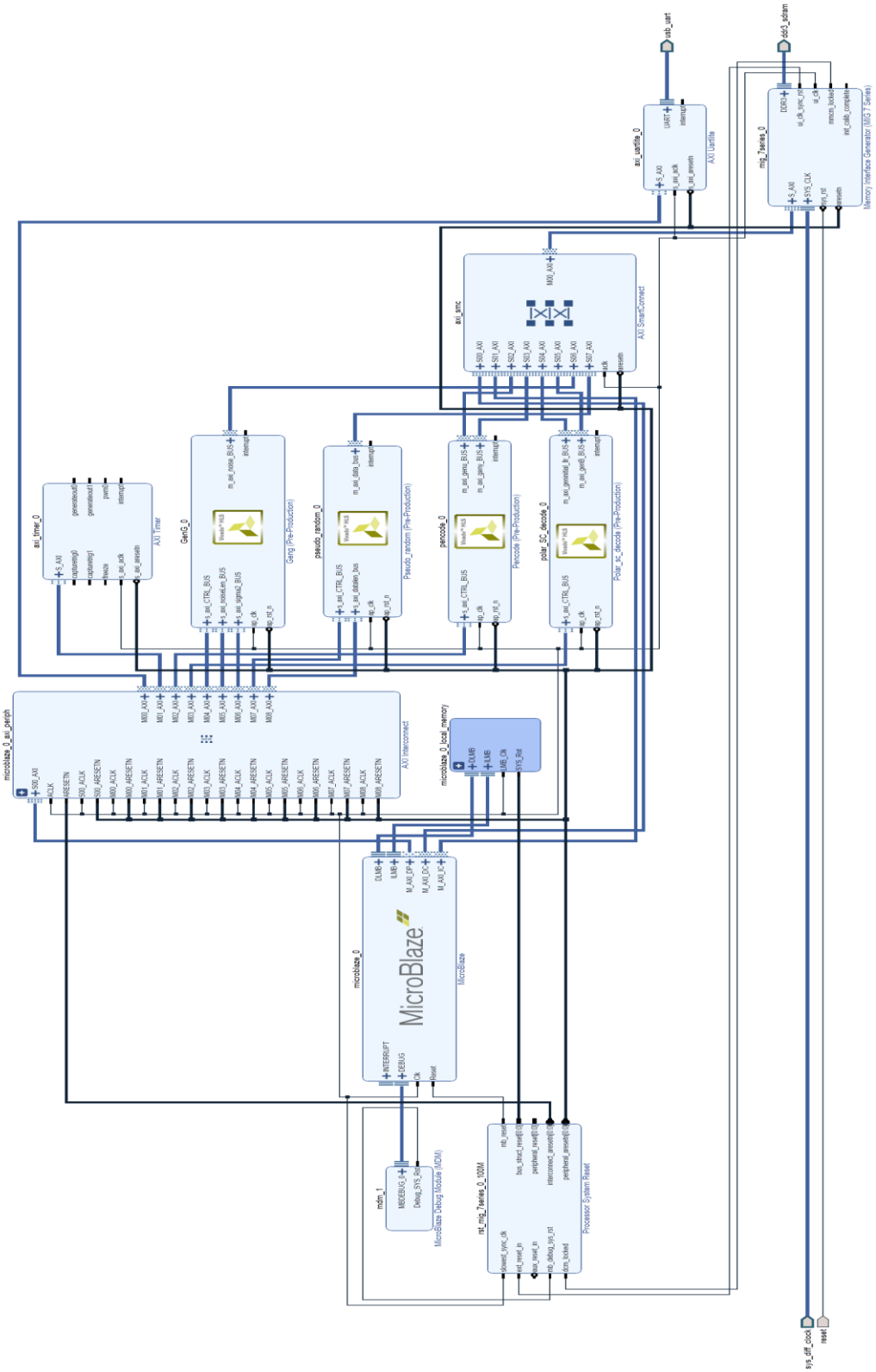


Figure (3.11): The Integrated Vivado Project.

3.5 Configuration of MicroBlaze Via Xilinx SDK

After generating the bitstream file using Xilinx Vivado, the hardware wrapper and the generated bitstream files will be exported to the SDK software, attached to the Vivado platform, for MicroBlaze programming. This is done using C/C++ language. The MicroBlaze is used to control the process of implementing the communication system. It utilizes the drivers of the IPs that are produced by Vivado and its associated base addresses.

The SDK linked the work parts to the MicroBlaze so that the flow of operations is controlled by programming the MicroBlaze using the main function in which a pre-defined address for each block was assigned. Algorithm B.1 which is shown in appendix B to implement the work completely, including generation binary data, polar encoding, adding Gaussian noise, demultiplexer and polar decoding on the board. To record the processing time of polar decoding, a timer is implemented to record the time through the decoding loop. The Bit Error Rate (BER), Frame Bit Error Rate (BER) and decoding time calculated and displayed using the USB-UART port provided by the axi-uartlite IP.

3.6 Procedure for Configuring FPGA

FPGA board is used to implement both the encoder and the decoder. In this method, we configuring the FPGA from the Vivado tool or SDK to download the bitstream file on board and display the results using the USB-UART port. Figure (3.12) illustrates the hardware arrangement for the proposed scheme.

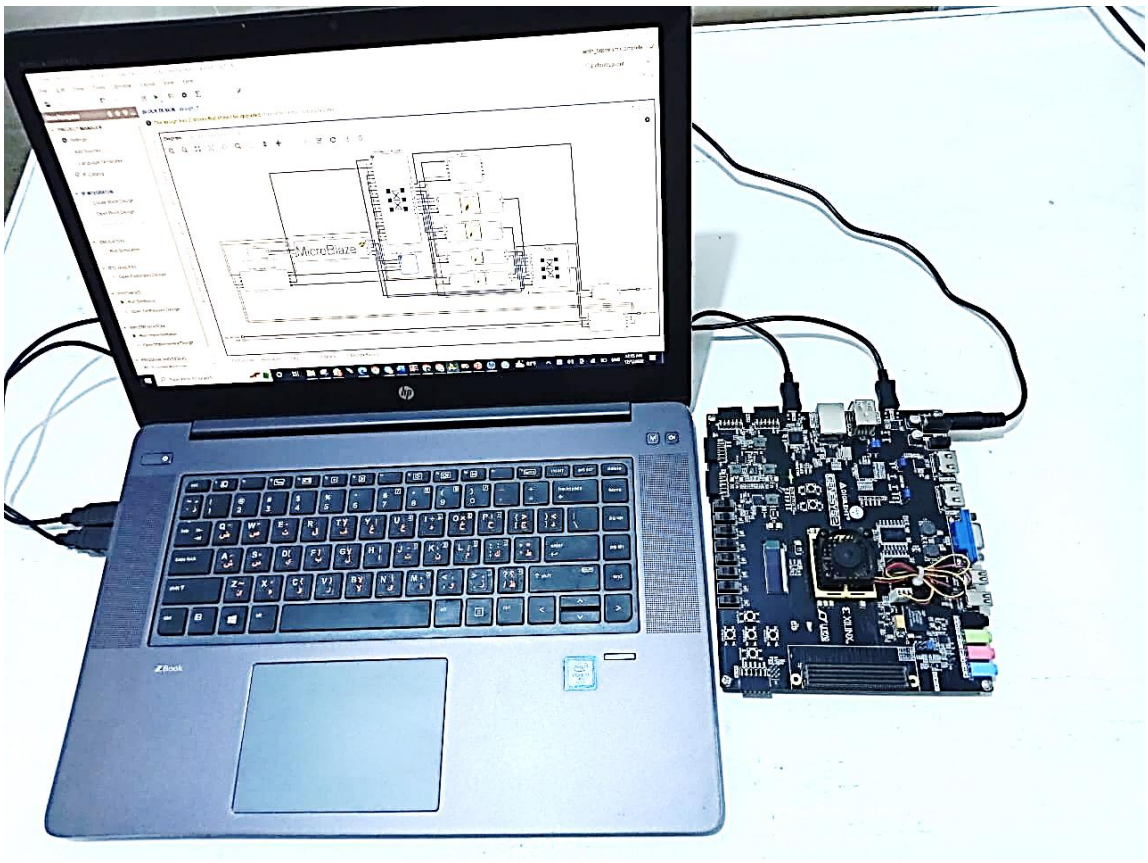


Figure (3.12): Configuration of FPGA board.

Chapter Four

Results and Discussion

4.1 Introduction

The results and discussions of the proposed polar codes for both simulation and hardware implementation are presented in this chapter. Intensive simulations are carried out, using Vivado HLS software version 2017.4 to show the importance of the proposed systems. Practical tests were made by manipulating Diligent Genesys 2 board which is an advanced, ready-to-utilize in addition to high performance digital circuit development platform based on a most recent Kintex7™ Field Programmable Gate Array with Xilinx part number XC7K325T2FFG900C.

The whole tested digital communication systems were collected using Vivado IP integrator. The comparison between different systems was based on calculating the Bit Error Rate (BER), Frame Error Rate (FER), silicon utilization, Power consumption estimation of the proposed implementation and estimated decoding time (latency).

4.2 The effect of employing different data types

A comparison in terms of BER performance of the system with 64,128 ,256 and 512 bits information length respectively and r=1/2 using the float, and ap_fixed<10,8,AP_TRN,AP_SAT>,ap_fixed<6,5,AP_TRN,AP_SAT>, ap_fixed <4,3,AP_TRN,AP_SAT> and ap-fixed<4,2,AP_TRN, AP_SAT>LLR are shown in Figures (4.1),(4.2),(4.3) and (4.4). Transforming LLR values passing through the polar decoder from float to a fixed format results in a degradation in BER performance.

The parameters of the polar codes that were modelled and developed with the help of the Vivado HLS and SDK platform are listed in Table (4.1).

Table (4.1): Polar encoder and decoder parameters.

Component of the encoder	$G = B_N \mathcal{F}^{\otimes n}$ Where B_N is the bit reversal transposition matrix $\mathcal{F}^{\otimes n}$ n-th Kronecker product of $\mathcal{F}$ $\mathcal{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$
Construction method	Gaussian
Code length	64,128,256,512,1024
Decoding Technique	Successive cancellation (SC)
Coding Rate	R=1/3,1/2, 2/3
Channel	AWGN

Fixed-point data types W I	Ap-fixed <W, I, Q, O> 10,6,4,4 8,5,3,2 where , W: stands for the hole's width in the signed word and takes up a single byte I: is fractional bits number Quantization and Overflow Modes are denoted by Q and O, respectively, and can take a variety of forms, as seen in [66]. Q: AP-TRN (Truncation to minus infinity) O: AP-SAT(Saturation)

The degree of this deterioration depends on the precision with which the data is represented. The simulation tests for different codeword lengths present an almost exact behavior of the system with data format “ap_fixed<10,8, AP_TRN, AP_SAT>” compared to the system utilizing float format. For the two systems with W=4, the one with fractional bits number I=3 presents an improvement of about 0.8 dB at BER of 10^{-4} over the one with I=2 for different code lengths. This reveals that fraction bits have a considerable effect on the decoding performance. As expected, when the codeword length increases, the performance will improve. This is mainly due to the increase in the minimum distance of the coded system. Generally, an improvement of about 0.5 dB is gained as the code length doubles.

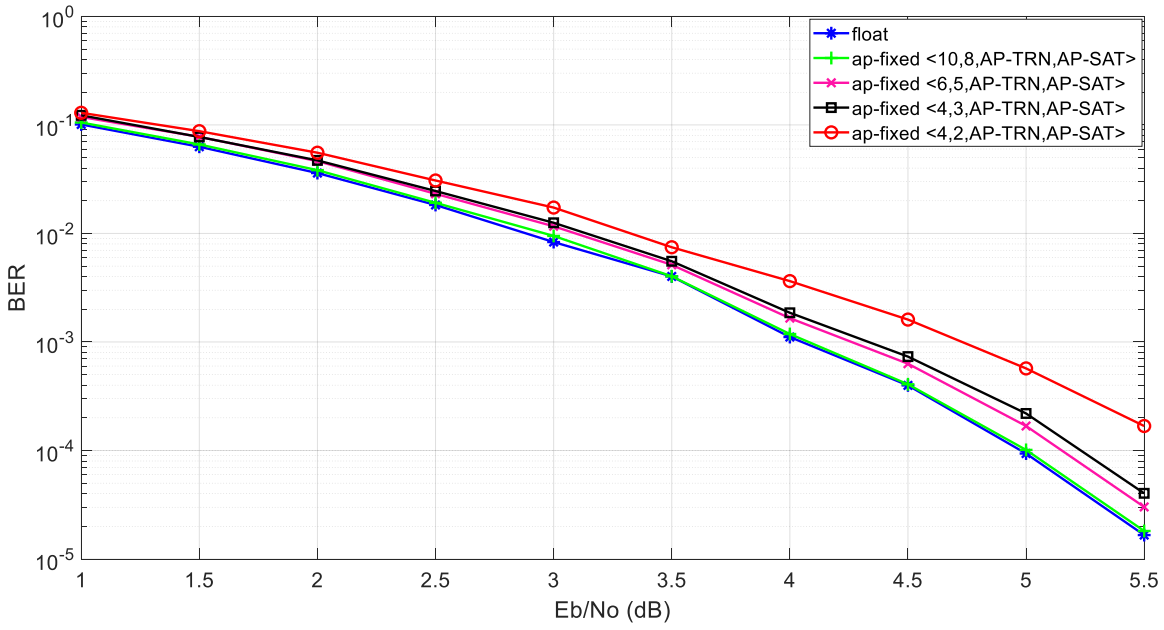


Figure (4.1): BER simulation performance for float and different arbitrary-precision (ap_ fixed) data types with r=1/2 and N=64.

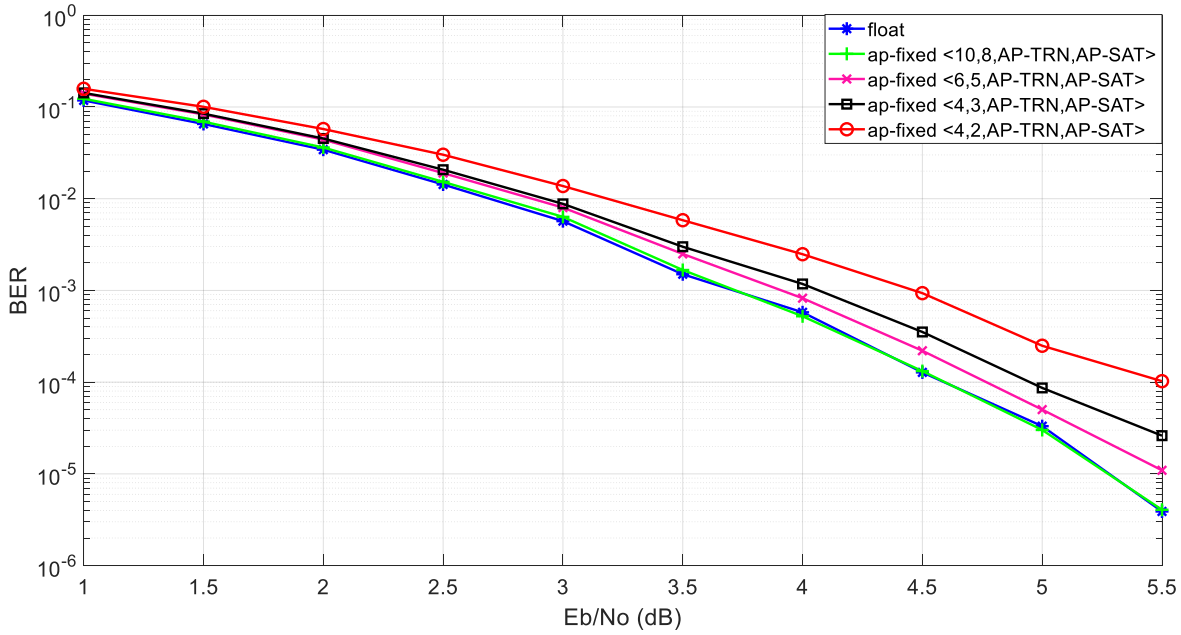


Figure (4.2): BER simulation performance for float and different arbitrary-precision (ap_ fixed) data types with r=1/2 and N=128.

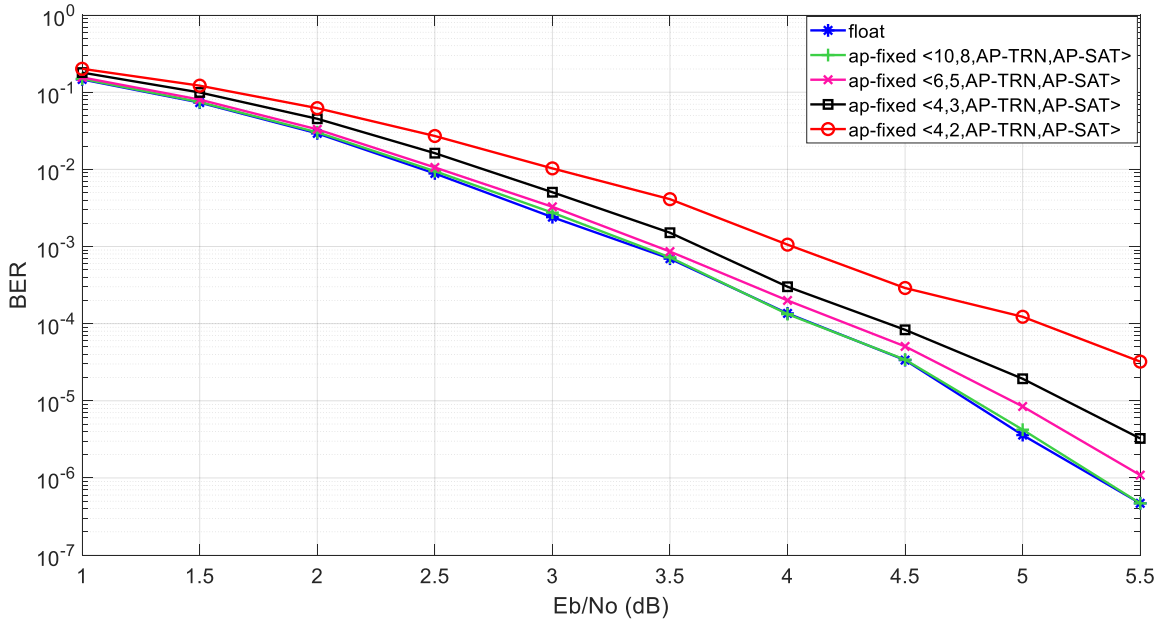


Figure (4.3): BER simulation performance for float and different arbitrary-precision (ap_ fixed) data types with r=1/2 and N=256.

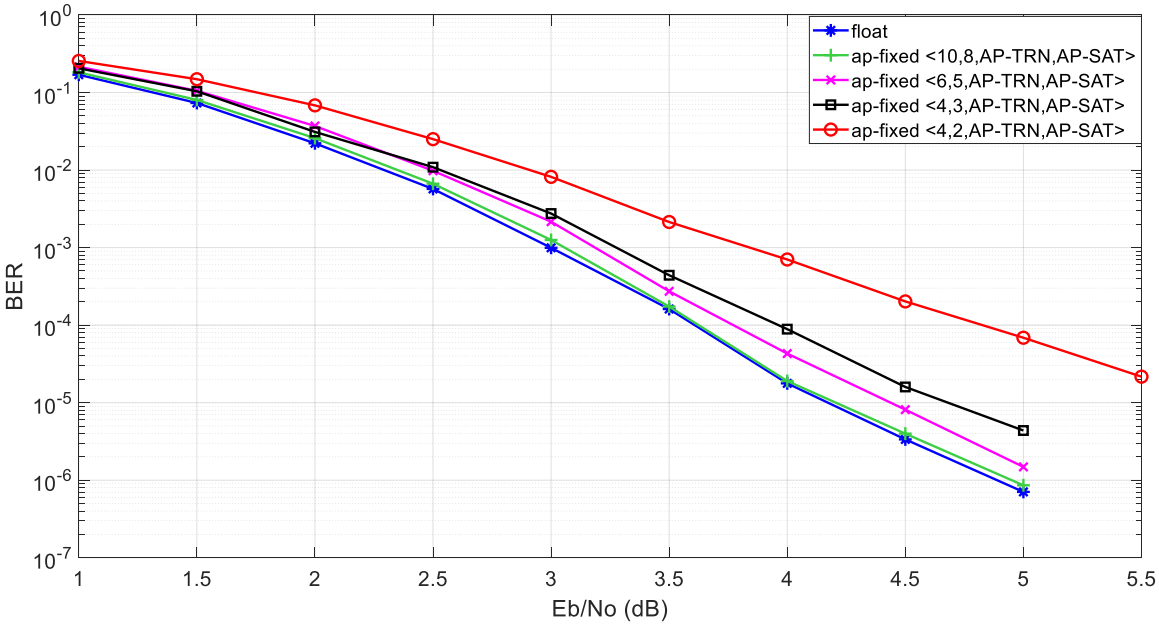


Figure (4.4): BER simulation performance for float and different arbitrary-precision (ap_ fixed) data types with r=1/2 and N=512.

Figure (4.5) show that the BER and FER performance of this study and prior study [19,21].

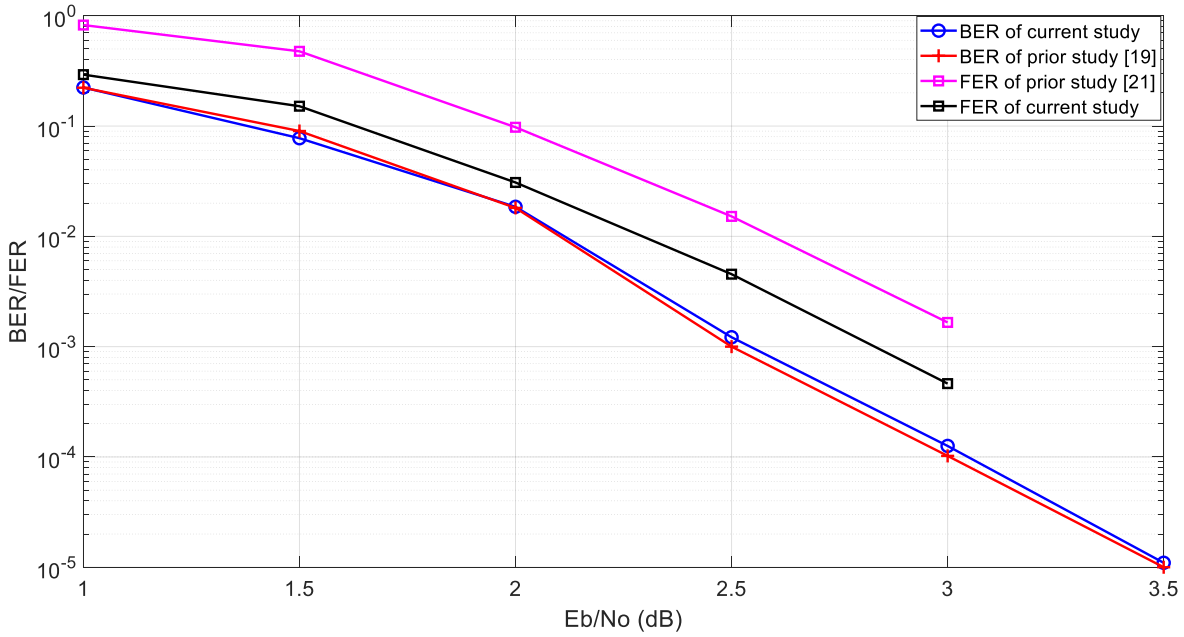


Figure (4.5): BER and FER performance of this study and prior study for N=1024, r=1/2 and float LLR data types.

Resources for proposed decoder and prior study for N= 64 polar codes [18] are shown in table (4.2).

Table (4.2) Resources comparison between proposed decoder and [18].

Resources	Proposed decoder	[18]
LUT	6958	9733
FF	3378	3003

Table (4.3) show the resources for proposed decoder and prior study for N= 1024 polar codes.

Table (4.3) Resources comparison between proposed decoder and the decoder of prior study [21,23].

Resources	Proposed decoder	[23]	[21]
LUT	6346	22115	190127
FF	3434	7941	22928

4.3 The effect of applying different HLS directives

In Vivado HLS, directives like array partitioning, pipelining, data-flow, and unrolling processes are employed. These directives have an impact on the hardware designs, and as a result, they have a significant impact on both performance and resource consumption. Usually, HLS provides different reports to the designer, which contains a lot of information about latency, initiation interval (throughput), and different FPGA resources (LUT, FF, BRAM, and DSP cores) that the design is consumed. The latency is defined by the number of clock-cycles wanted for an IP (generated by a function) to calculate all values of output. Whereas, Initiation-Interval (II) is defined by the number of clock-cycles (given that one cycle of the clock equals ten nanoseconds) before the function can call for new information (data of input).

The details presented in Table (4.5) pertain to latency (expressed in terms of the number of clock cycles), as well as the various FPGA resources (LUT, FF, BRAM, and DSP processor) that are utilized by the SC decoder four synthesis solutions that utilize various optimization directives as shown in table (4.4).

Table (4.4) Description of the solutions

No.	Solution description
Solution1	It makes use of the standard HLS directive, which is configured in such a way as to make a balance between throughput and the amount of resources required
Solution2	It keeps the standard HLS directive from solution1 and adds two optimizations is added to the code: 1- (hls::fmod ((float)psi,(float) 2) != 0) is replaced by (Y & 0x1) != 0) 2- Y = hls::floor(X / 2.) is replaced by Y = X/2
Solution3	it adds a variety of parallelism directives to solution 2 such as pipeline, loop_tripcount and partitioning all arrays.
Solution4	In this solution of arbitrary-precision data types, specifically <ap-fixed10,8, AP TRN, AP SAT> are used. The LLR data type, which denotes a fixed-point number, has a significant impact on both latency and resource consumption.

The deterioration in performance for systems using fixed data types is compensated by the reduction in resource utilization and latency as shown in Table (4.5). This is mainly due to the fewer memory and processing needed for fixed type. This table also shows that using different HLS directives can reduce the delay time on the account of increasing utilization resources and vice versa.

Table (4.5) An estimate of the SC decoder's latency and utilization for N=256 and R=0.5.

Resources	solution1	solution2	solution3	solution4
Latency (cycles)	52238	14349	14863	12566
BRAM_18K	33	22	22	6
DSP48E	43	8	8	6
FF	8581	3491	3631	3996
LUT	19201	6662	6998	9886

Table (4.6) and (4,7) illustrate the resource utilization and latency of the SC decoder with different arbitrary-precision fixed types and N=256,512. It is evident from this Table that low precision data type in solution 4 consumed fewer. Likewise, the latency in this solution is greatly reduced.

Table (4.6) The estimated latency and utilization for various arbitrary precision data types for N=256 and R= 1/2.

Resources	Ap-fixed<10,8,AP-TRN,AP-SAT>	Ap-fixed<6,5,AP-TRN,AP-SAT>	Ap-fixed<4,3,AP-TRN,AP-SAT>	Ap-fixed<4,2,AP-TRN,AP-SAT>
Latency	12566	12566	12566	12566
BRAM_18K	8	7	6	6
DSP48E	6	6	6	6
FF	4046	3964	3948	3996
LUT	10252	10096	10026	9886

Table (4.7) The estimated latency and utilization for various arbitrary precision data types for N=512 and R= 1/2.

Resources	Ap-fixed<10,8,AP-TRN,AP-SAT>	Ap-fixed<6,5,AP-TRN,AP-SAT>	Ap-fixed<4,3,AP-TRN,AP-SAT>	Ap-fixed<4,2,AP-TRN,AP-SAT>
Latency	27158	27158	27158	27158
BRAM_18K	10	8	7	7
DSP48E	6	6	6	6
FF	4126	4044	4028	4076
LUT	10321	10165	10095	9955

4.4 The effect of applying different data rates

Different simulation tests with different data rates are carried out for float LLR data type and polar-coded systems with a length of 64 and 256. In general, systems with low coding rates present better performance. A low rate means more added redundancy and hence larger distance spaces between adjacent codewords (maximizing minimum Hamming distance). Code with maximum Hamming distance performs well due to its higher error correction capability. This effect is illustrated in Figures (4.6) and (4.7).

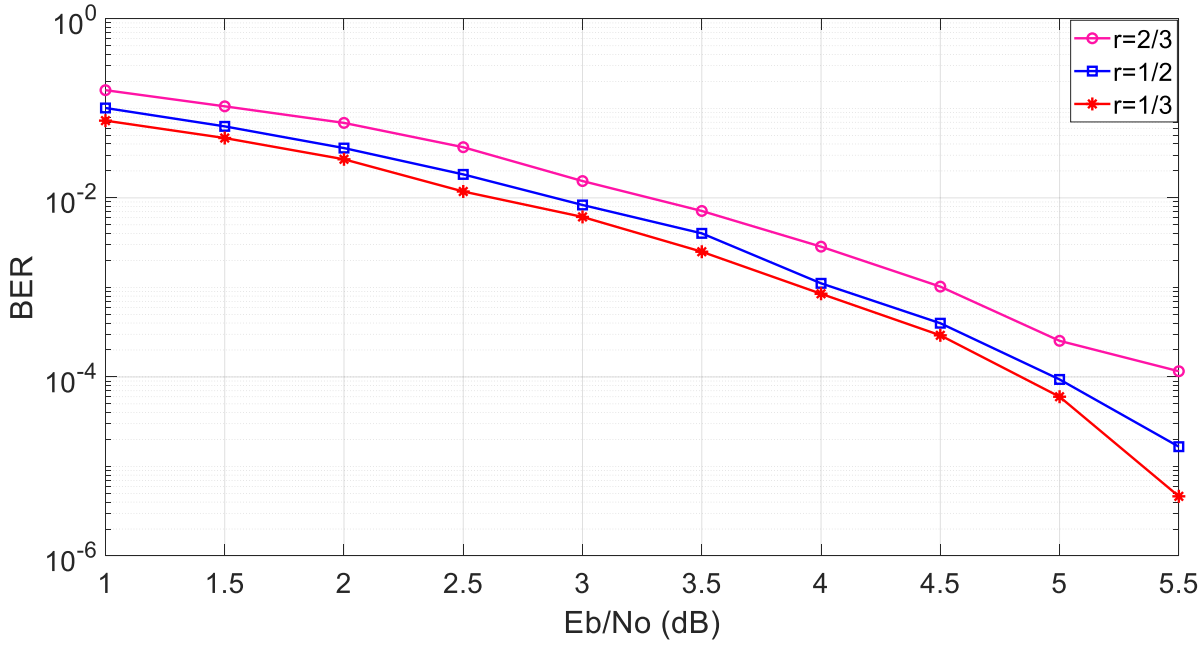


Figure (4.6): BER of simulation with different rates and length of 64 bits and using float LLR data type.

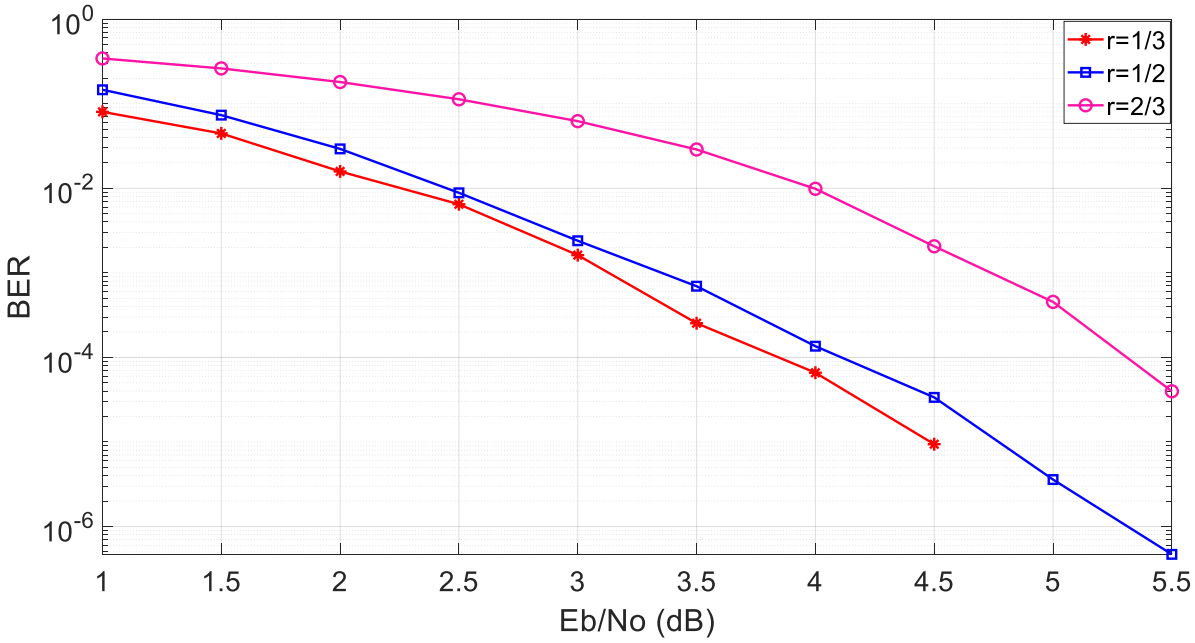


Figure (4.7): BER performance with different rates and lengths of 256, using float LLR data type.

4.5 The effect of utilizing different polar code lengths

The effect of frame length is illustrated in Figure (4.8), where five lengths 64,128, 256,512 and 1024 bit are simulated, all with r=1/2. Different simulation tests are carried out for polar-coded systems and float LLR data types. The performance is improved by increasing the frame size, and a good BER obtains. This is due to the improvement in the distance spectrum of the code.

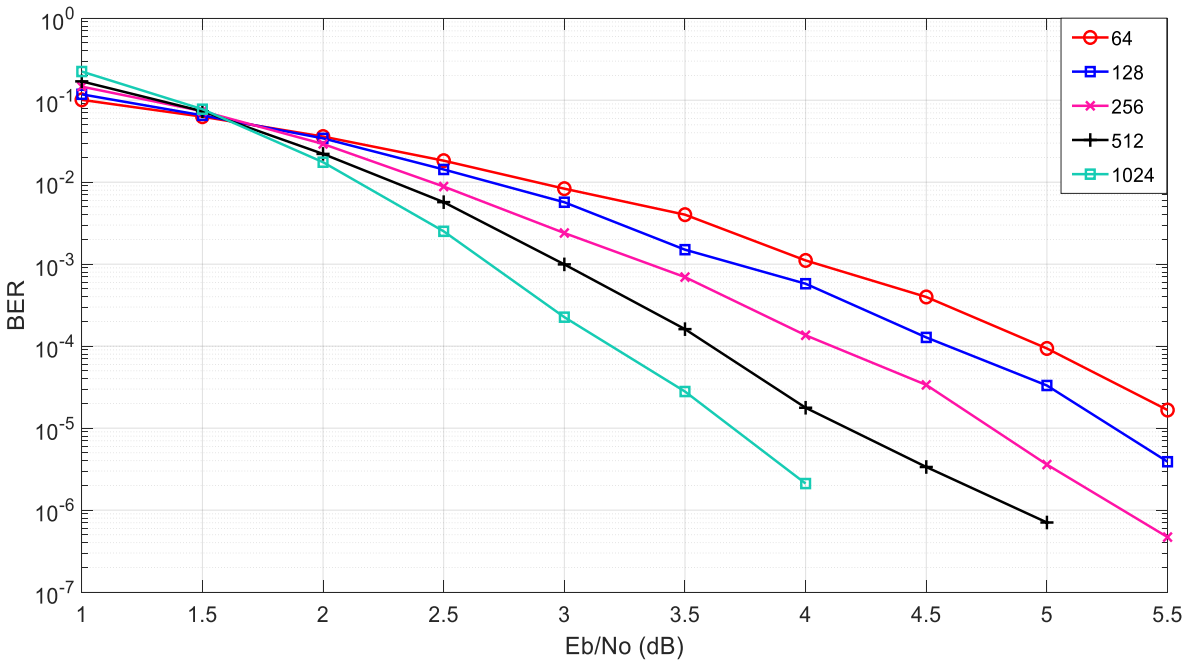


Figure (4.8): BER of simulation with different length, r=1/2 and using float LLR data type.

Figure (4.9), (4.10) and (4.11) shows the BER when different frame sizes lengths 64,128, 256,512 and 1024 bit/frame are used at r=1/2 and ap-fixed LLR data type. As shown in these figure that the codes with larger polar code length have improved performance. This is due to the improvement in the distance properties of the code.

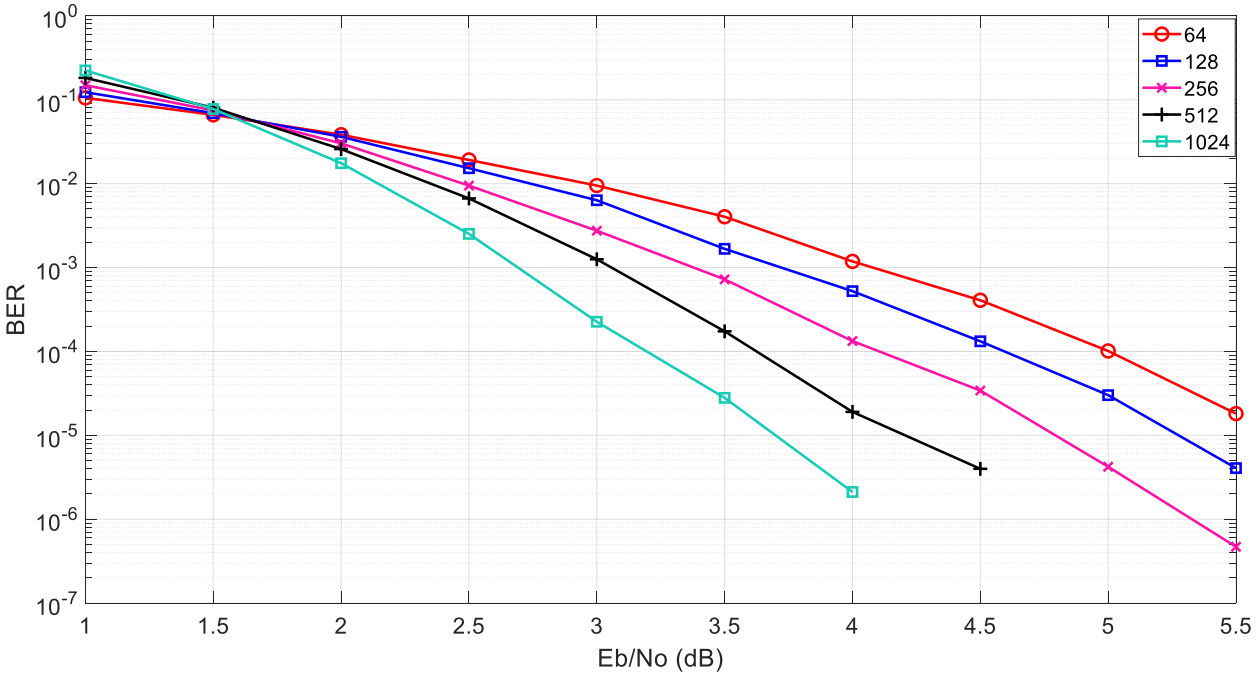


Figure (4.9): BER of simulation with different length, r=1/2 and using ap-fixed <10,8, AP_TRN, AP_SAT> data type.

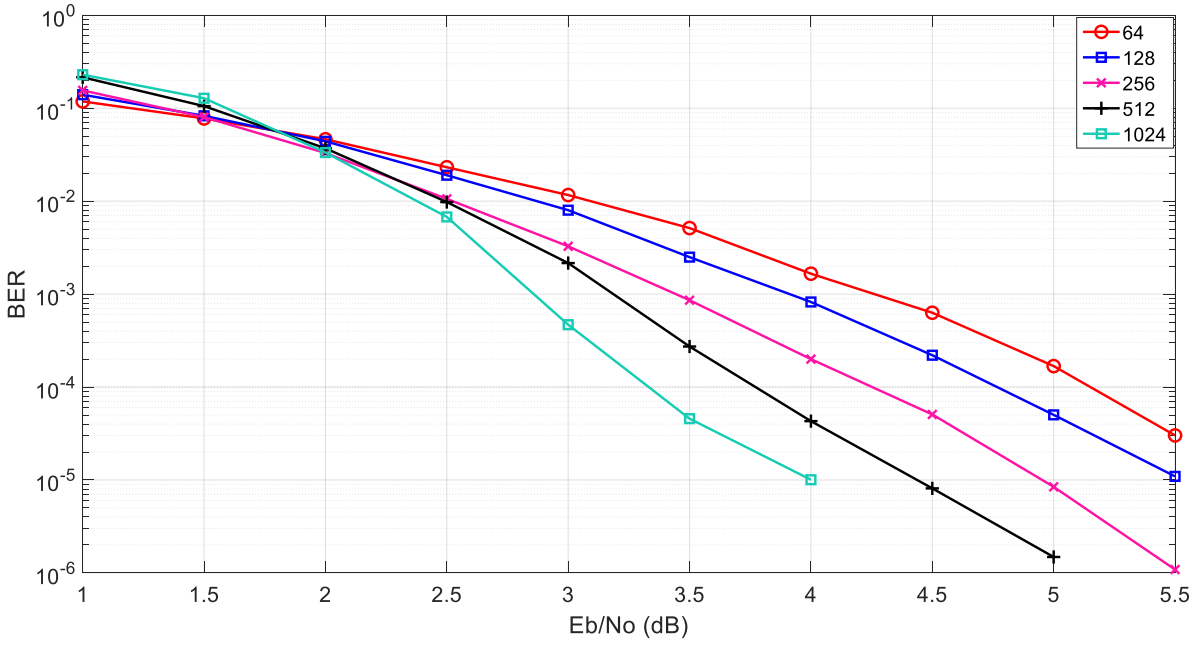


Figure (4.10): BER of simulation with different length, r=1/2 and using ap-fixed <6,5, AP_TRN, AP_SAT> data type.

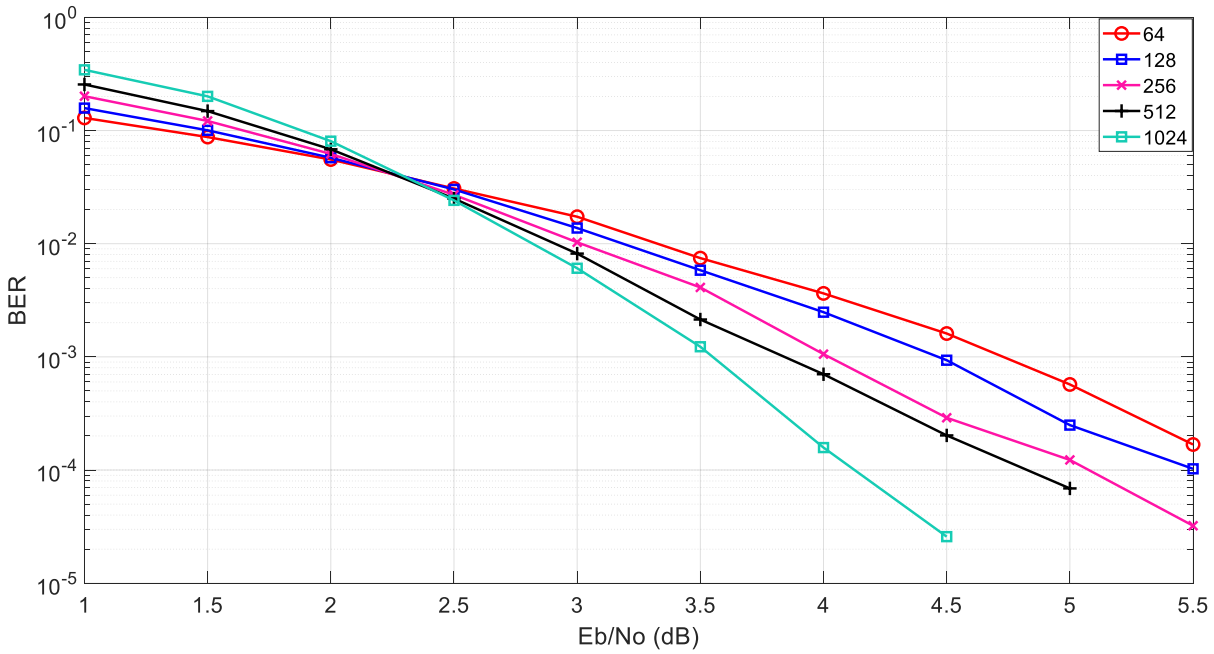


Figure (4.11): BER of simulation with different length, r=1/2 and using ap-fixed <4,2, AP_TRN, AP_SAT> data type.

4.6 Hardware results

Different hardware tests are carried out for polar code system, Figures (4.12),(4.13),(4.14) and (4.15) show a comparison in terms of BER performance of the system with 64,128 ,256 and 512 bits information length respectively and r=1/2 using the float, and ap_fixed<10,8,AP_TRN,AP_SAT>,ap_fixed<6,5,AP_TRN,AP_SAT>, ap_fixed <4,3,AP_TRN,AP_SAT> and ap-fixed<4,2,AP_TRN, AP_SAT>LLR.BER performance begins to decrease when float LLR values that pass through the polar decoder are changed to fixed format. How much this happens depends on how precise the data is. Simulation tests for different codeword lengths show that the system with data format "ap fixed<10,8,AP TRN,AP SAT>" behaves almost exactly like the system that uses float format. For the two systems with W=4, the one with fractional bits number I=3 is better than the one with I=2 for different code lengths by about 1.0 dB at

BER = 10^{-4} . This shows that fraction bits have a big effect on how well the decoding works. As expected, the performance will get better as the length of the codeword grows. This is mostly because the minimum distance for the coded system has increased. Most of the time, doubling the length of the code improves the outcome by about 0.5 dB.

In the absence of arbitrary precision data types, multiplication must be implemented using 32-bit integer data types, leading to the need for numerous DSP and other modules to carry out the operation. Figure (4.16) show the FER hardware performance for float and different arbitrary-precision (ap_ fixed) data types with r=1/2 and N=256.

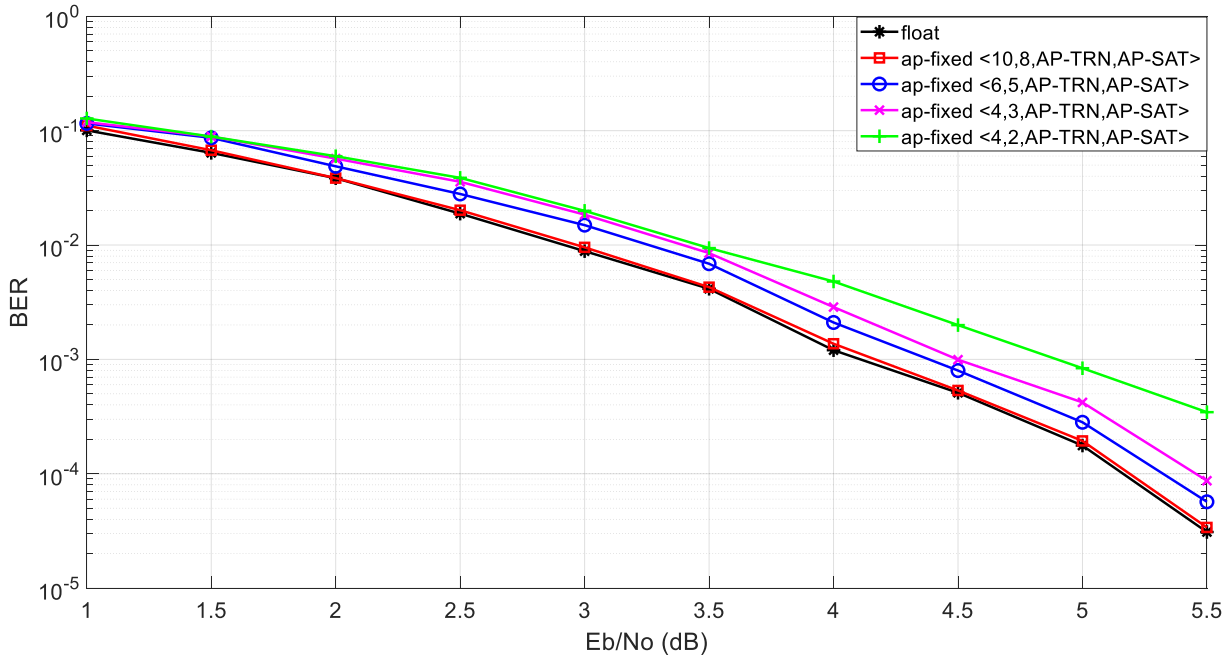


Figure (4.12): BER hardware performance for float and different arbitrary-precision (ap_ fixed) data types with r=1/2 and N=64.

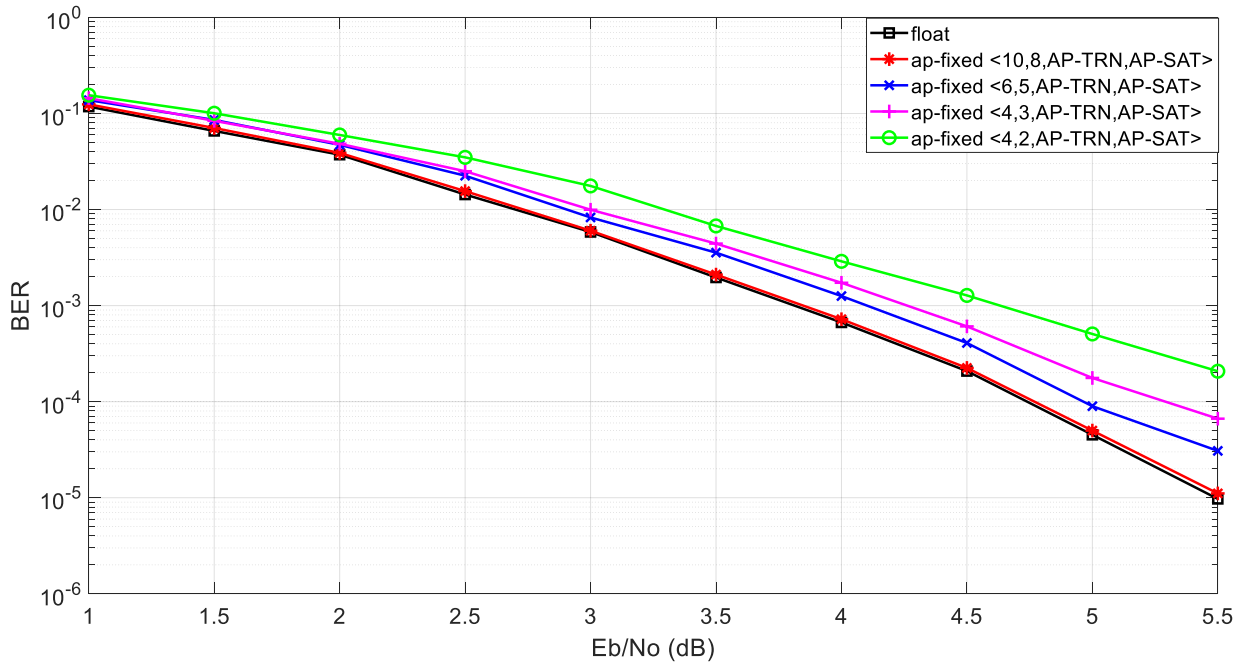


Figure (4.13): BER hardware performance for float and different arbitrary-precision (ap_ fixed) data types with r=1/2 and N=128.

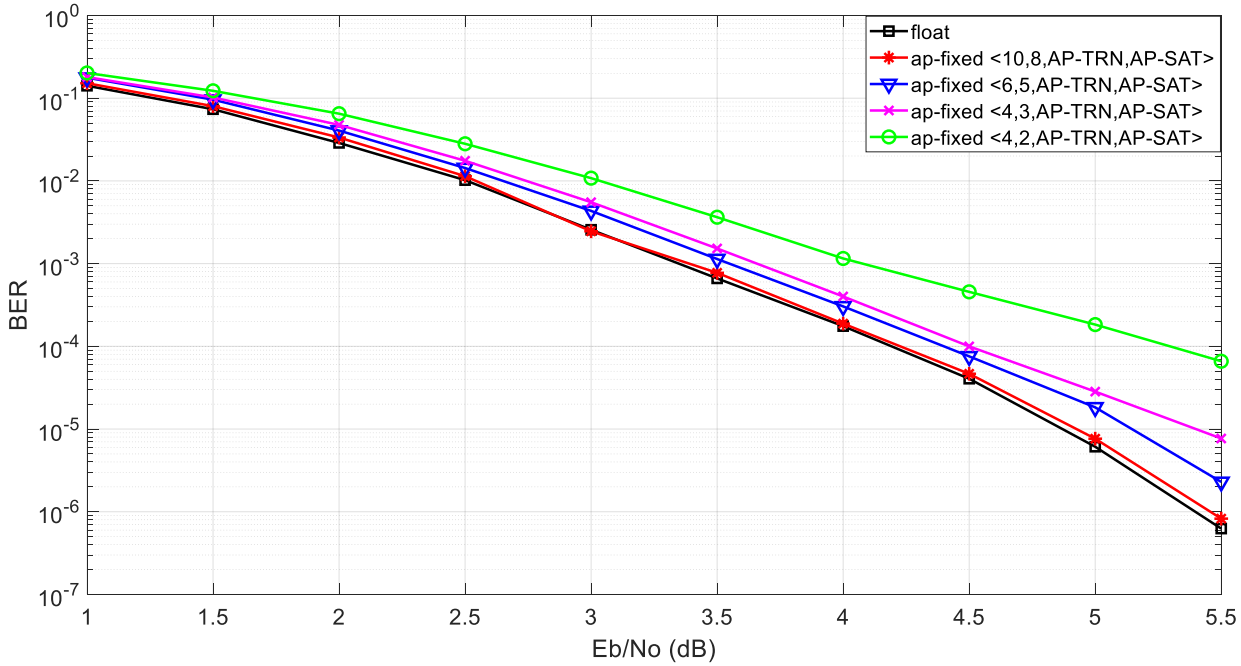


Figure (4.14):): BER hardware performance for float and different arbitrary-precision (ap_ fixed) data types with r=1/2 and N=256.

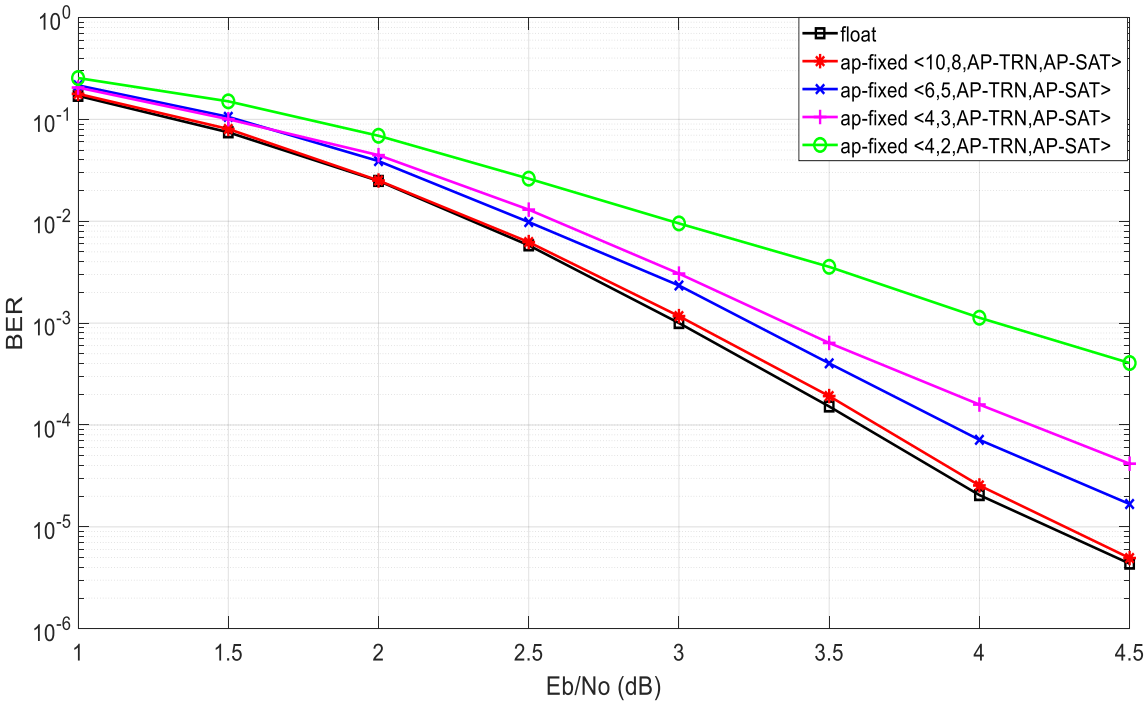


Figure (4.15): BER hardware performance for float and different arbitrary-precision (ap_ fixed) data types with r=1/2 and N=512.

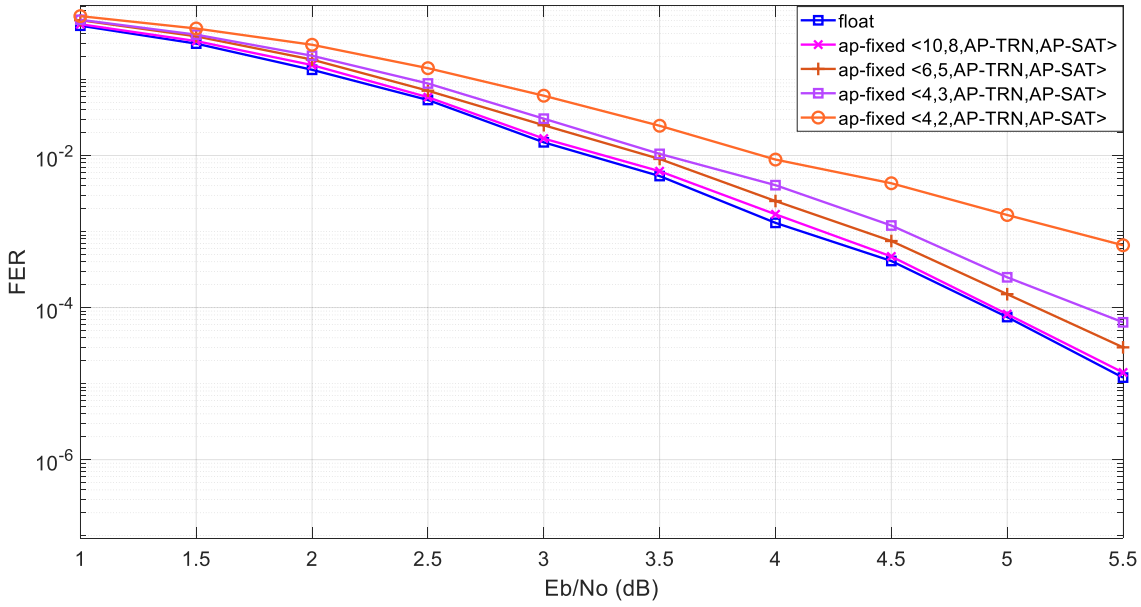


Figure (4.16): FER hardware performance for float and different arbitrary-precision (ap_ fixed) data types with r=1/2 and N=256.

4.7 Simulation and Hardware Comparison

Comparing the hardware and software results is interesting to increase the confidence between these acquired results. Figures (4.17), (4.18), (4.19), (4.20), (4.21), (4.22), and (4.23) show the BER comparison of the hardware and simulation results of the polar coded system of length 64, 128, 256 and 512, r=1/2 and ap-fixed LLR data type. From these figures, it is demonstrated that the hardware constraint of the FPGA device causes the simulated system to outperform the hardware performance with a range between 0.1 dB to 0.5 dB at BER of 10^{-4} .

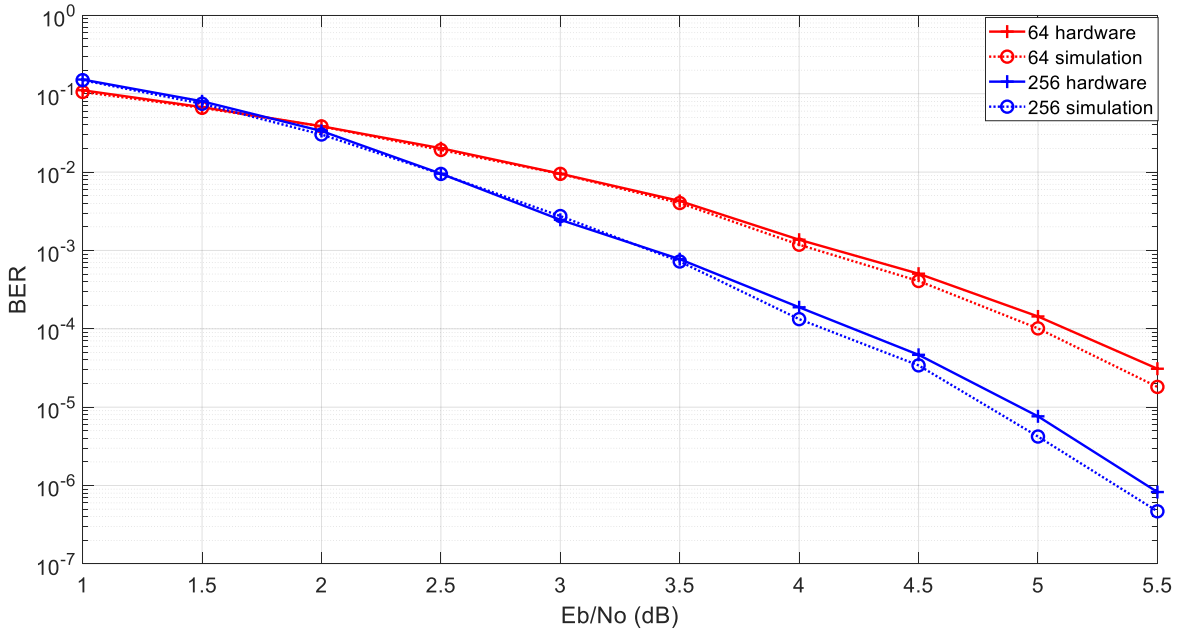


Figure (4.17): BER performance of hardware and simulation for r=1/2, N=64 and 256 using ap-fixed <10,8, AP_TRN, AP_SAT> data type.

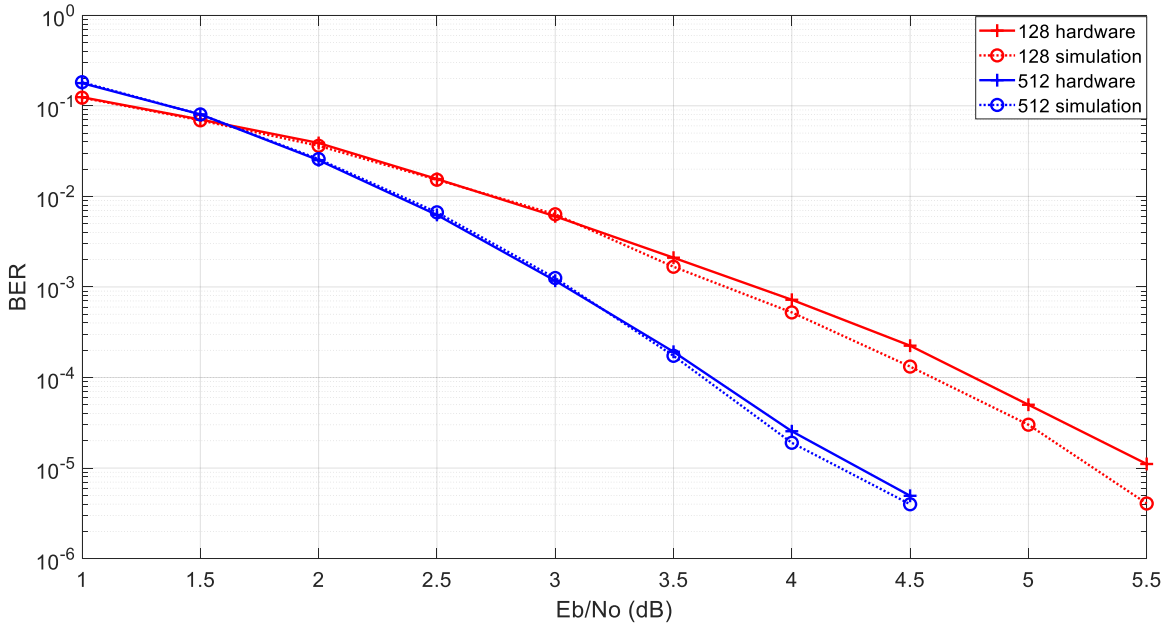


Figure (4.18): BER performance of hardware and simulation for r=1/2, N=128 and 512 using ap-fixed <10,8, AP_TRN, AP_SAT> data type.

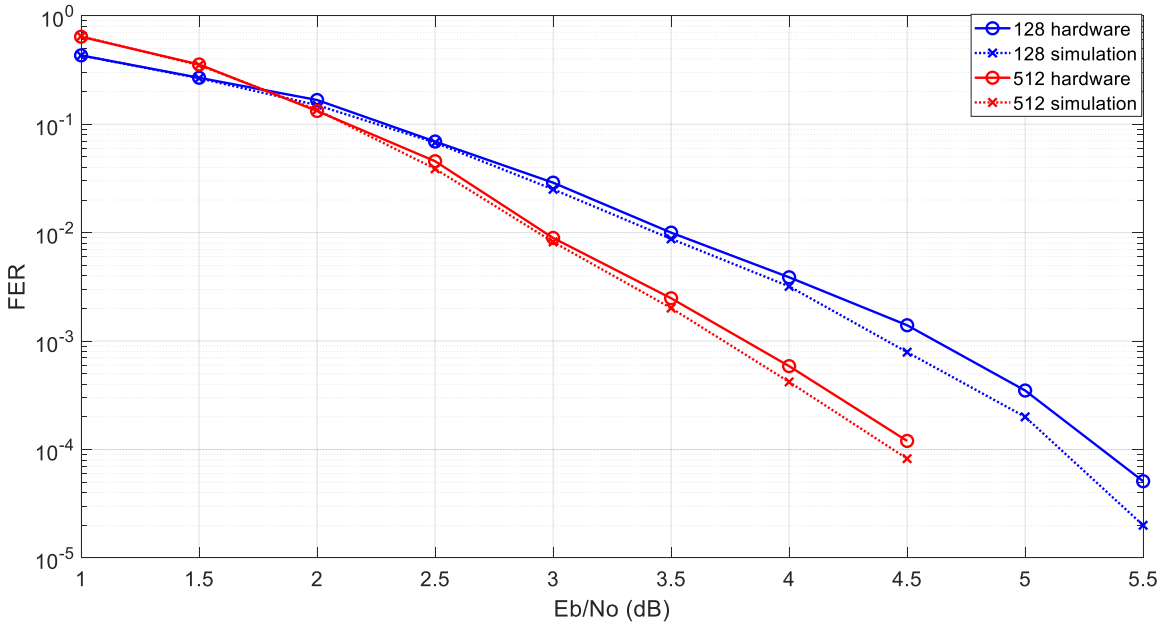


Figure (4.19): FER performance of hardware and simulation for r=1/2, N=128 and 512 using ap-fixed <10,8, AP_TRN, AP_SAT> data type.

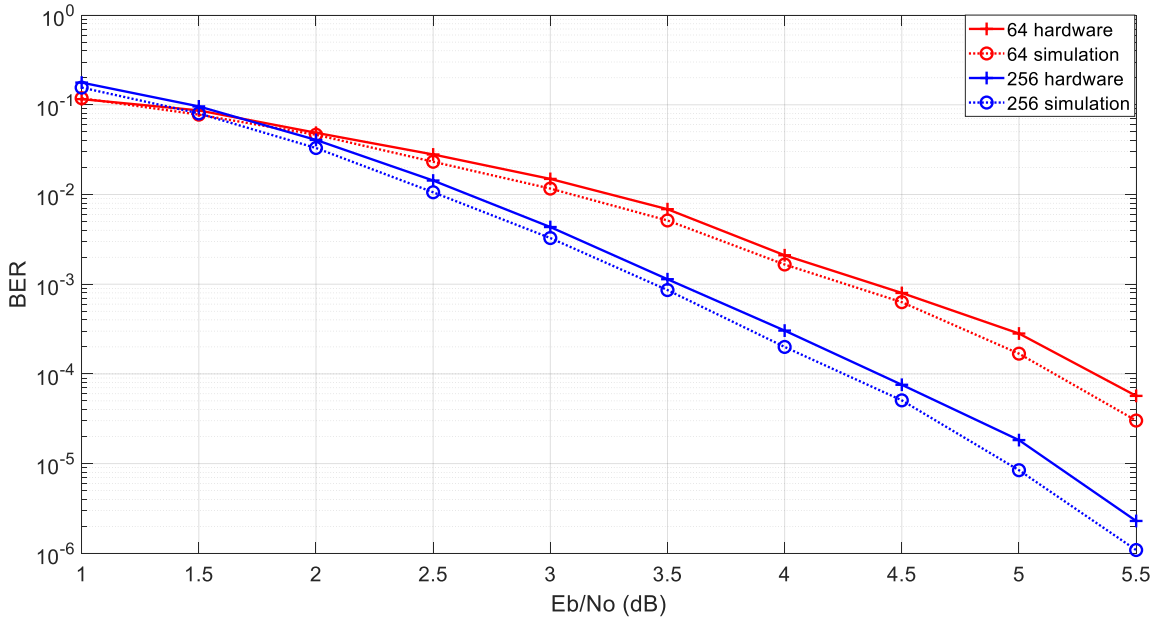


Figure (4.20): BER performance of hardware and simulation for r=1/2, N=64 and 256 using ap-fixed <6,5, AP_TRN, AP_SAT> data type.

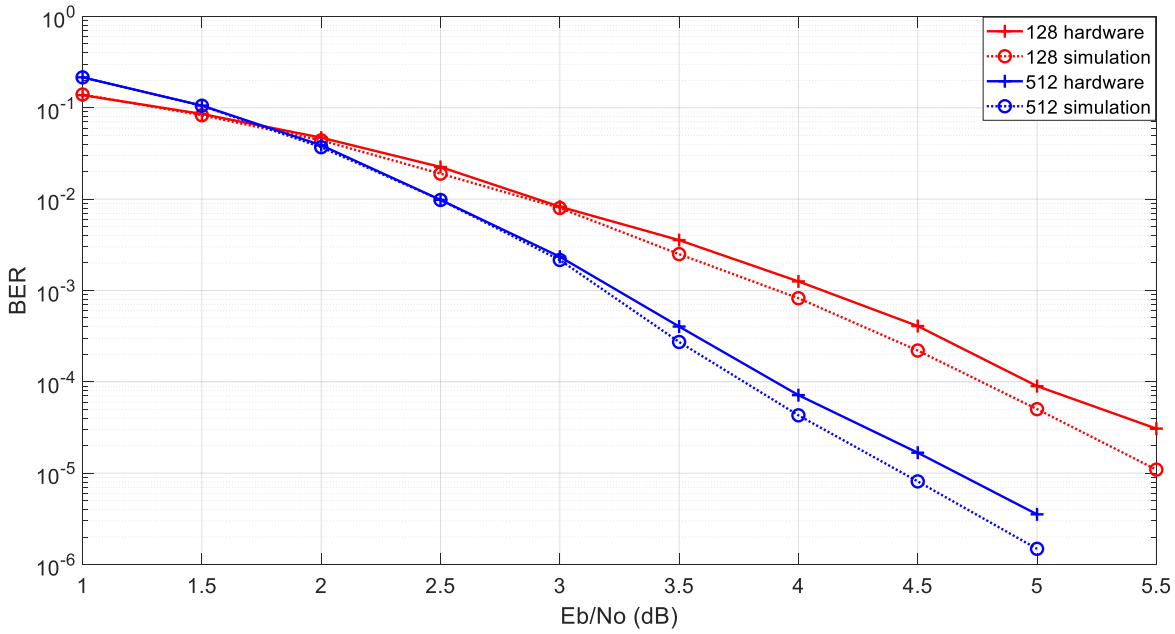


Figure (4.21): BER performance of hardware and simulation for r=1/2, N=128 and 512 using ap-fixed <6,5, AP_TRN, AP_SAT> data type.

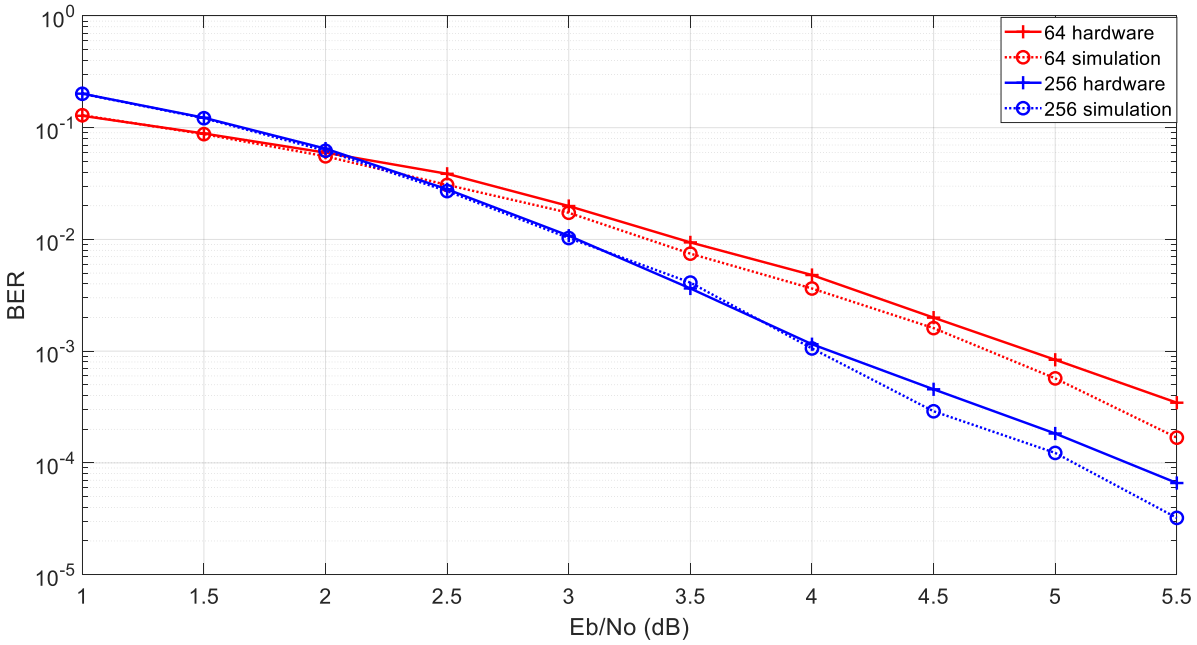


Figure (4.22): BER performance of hardware and simulation for r=1/2, N=64 and 256 using ap-fixed <4,2, AP_TRN, AP_SAT> data type.

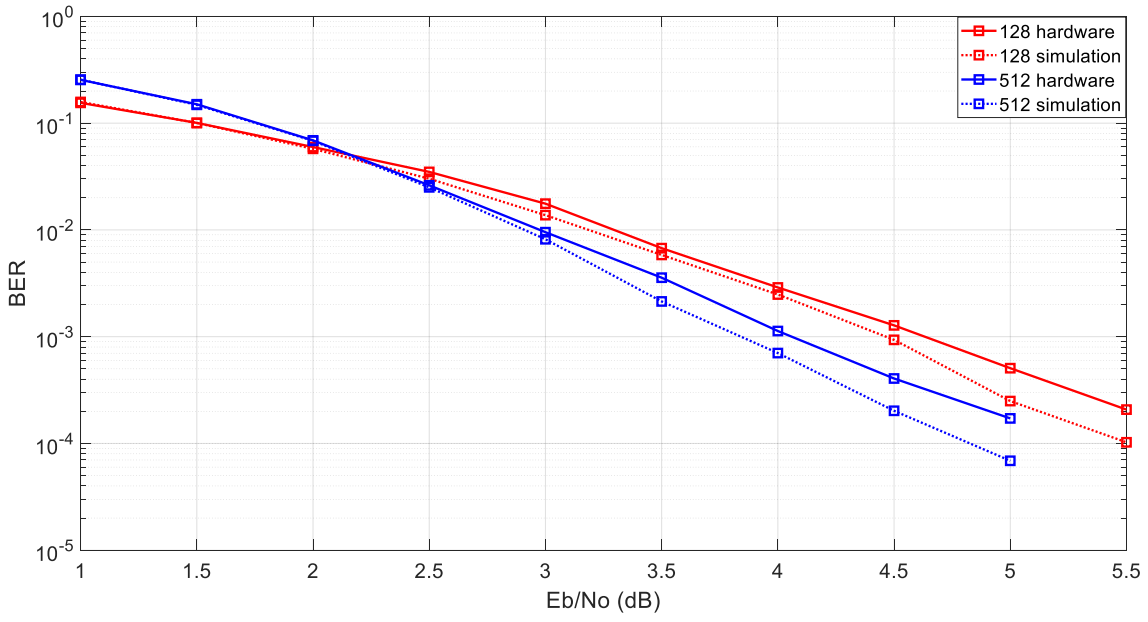


Figure (4.23): BER performance of hardware and simulation for r=1/2, N=128 and 512 using ap-fixed <4,2, AP_TRN, AP_SAT> data type.

4.8 Simulation and hardware time comparison

The comparison of latency between the hardware and simulation results of the polar coded systems of different code lengths is presented in figure (4.24). The timing results show an increase in latency by 50% for hardware. This is mainly due to the reading and writing operations of the input and output arguments of the IP decoder block from and to DDR RAM.

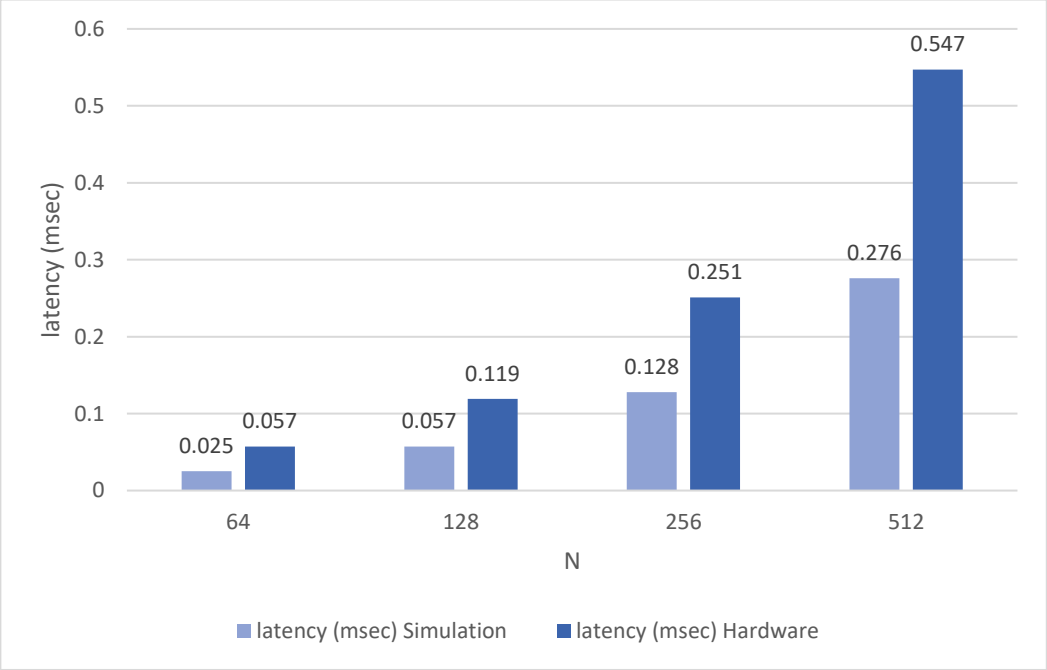


Figure (4.24): Latency of hardware and simulation of different polar code lengths, r=1/2 and using float data type.

4.9 Resources utilization and power consumption result

figure (4.25) depicts the utilization report provided by Vivado IP integrator that shown in figure (3.11) of 256 polar codes with ap_fixed <10,8, AP_TRN, AP_SAT> data types.

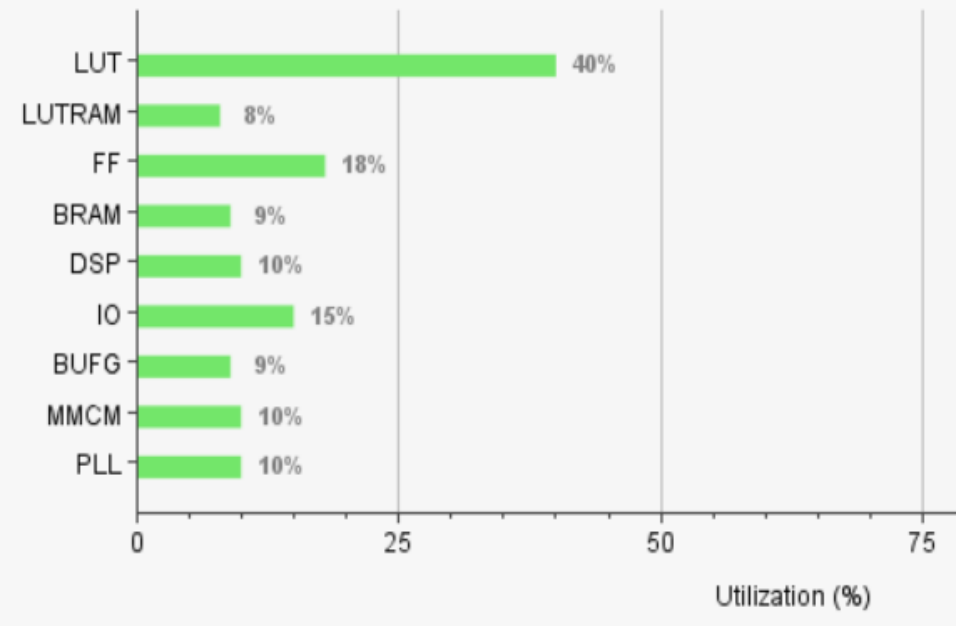


Figure (4.25): Resources utilization of the proposed implementation system which is shown in figure (3.11) of N= 256 for ap_fixed <10,8, AP_TRN, AP_SAT> LLR data type.

Based on the utilization report in Figure (4.25), 40 (%) of the LUTs, 10(%) of the DSP48E, (9%) of the BRAM, and 18 (%) of the FFs are used to implement the system. IP of the decoder use most of the resources because of its complexity as compared with the another Ips.

The dynamic and static power consumed by Vivado IP integrator of 256 is given in Fig (4.26).

The details of the estimated power consumption of the implementation are summarized in Fig (4.26). The I/O consumes more power than the others (about

28%), and DSP and BRAM consume only a small portion of the total on-chip power consumption (about 4%). The total on-chip power consumption estimation for the implementation is 2.086, which includes 1.910 W dynamic power and 0.176 W static power

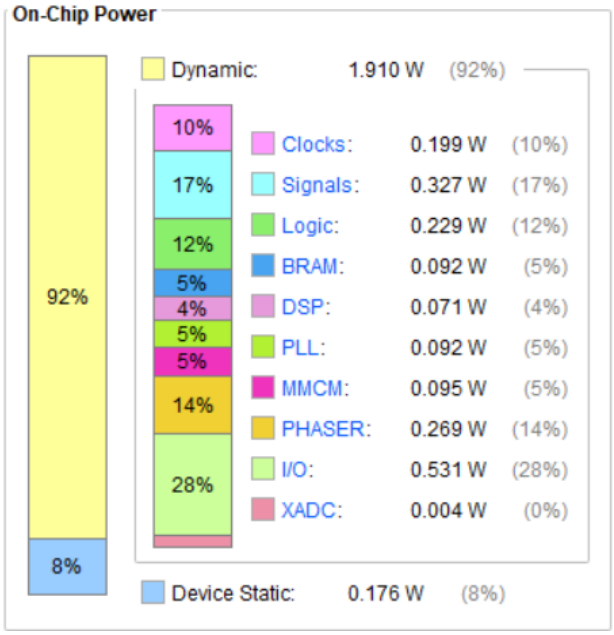


Figure (4.26): Power consumption estimation of the proposed implementation system for N=256 with ap_fixed<10,8, AP_TRN, AP_SAT> LLR data type.

Chapter Five

Conclusions and Suggestions for Future Works

5.1 Conclusions

The performance of various SC based polar decoding schemes in the context of the NR standard that used in 5G was investigated. Different simulation and hardware implementation are carried out in this thesis to improve the BER performance, latency, throughput, silicon area, and hence power consumption. The findings show that using different directives and low-precision data types can reduce the latency and utilization without significantly reducing BER performance. The following point will conclude the results achieved.

1. The ability of Vivado HLS in applying various directives like pipelining, unrolling, array partition, etc., it is possible to make a tradeoff between the performance (throughput and latency) and the required silicon area for the designed IP's without any sacrifice in the BER, FER.
2. Using the Kintex 7 FPGA and the Vivado platform, a variety of hardware tests are performed in order to validate the findings of the simulations performed on a number of different polar coded systems. It was discovered that a substantial decrease in latency and utilization resources may be attained with just a slight decline in BER and FER performance.
3. Using a variety of fixed-point data types rather than the more expensive float type to represent the LLR information that is processed by the SC decoders allows for a considerable decrease in latency as well as a reduction in the amount of silicon area that is required.

5.2 Suggestions for Future Works

In this thesis, we designed and implemented a Polar Decoder based on Successive Cancellation. In this section we suggest some improvements for future work.

1. More efficient successive cancellation methods could be considered, such as SSC and FSSC. These two improved successive cancellation methods cost less time than the normal SC method but achieve the same performance.

2. More sophisticated FPGA devices may be utilized, like Kintex UltraScale, Zynq UltraScale, Virtex-7 etc., which offers hard processors instead of the MicroBlaze soft processor to improve the implementation of the designed IP cores.

3. The proposed system in this thesis is based on using Binary phase-shift keying (BPSK) signal over AWGN channel. Rayleigh, Rician fading channels and various modulation schemes with higher signal constellations can be investigated.

4. The polar codes have better error floor performance compared to Turbo codes and LDPC codes, as well as having a larger girth of 12 than those two other types of codes. Therefore, in terms of error performance, polar codes are a better contender for the FEC technology than turbo codes and LDPC codes for the use of fiber-optical communication and data storage. It seems like figuring out how to use polar codes to fiber-optic communication and the application of data storage might be a successful area of research.

References

[1] A. Imran, "**Software implementation and performance of UMTS turbo decoder**," Msc. thesis, Tampere University of Technology, 2012

[2] E. Arikan, "**Channel polarization: A method for constructing capacityachieving codes for symmetric binary-input memoryless channels**," IEEE Transactions on Information Theory, vol. 55, no. 7, pp. 3051–3073, 2009.

[3] K. Niu, K. Chen, J. Lin, and Q. Zhang, "**Polar codes: Primary concepts and practical decoding algorithms**," IEEE Communications magazine, vol. 52, no. 7, pp. 192–203, 2014.

[4] R. Umar, F. Yang, and S. Mughal, "**Ber performance of a polar coded ofdm over different channel models**," in Applied Sciences and Technology (IBCAST), 2018 15th International Bhurban Conference on. IEEE, 2018, pp. 764–769.

[5] V. Bioglio, C. Condo, and I. Land, "**Design of polar codes in 5g new radio**," arXiv preprint arXiv:1804.04389, 2018.

[6] A. Bhatia, V. Taranalli, P. H. Siegel, S. Dahandeh, A. R. Krishnan, P. Lee, D. Qin, M. Sharma, and T. Yeo, "**Polar codes for magnetic recording channels**," in 2015 IEEE Information Theory Workshop (ITW). IEEE, 2015, pp. 1–5.

[7] U. U. Fayyaz and J. R. Barry, "**Polar code design for intersymbol interference channels**," in Global Communications Conference (GLOBECOM), 2014 IEEE. IEEE, 2014, pp. 2357–2362.

[8] M. Oda and T. Saba, "**Polar coding with enhanced channel polarization**

under frequency selective fading channels," in 2018 IEEE International Conference on Communications (ICC). IEEE, 2018, pp. 1–6.

[9] R. Mahdi, and A. Hamad. "**Implementation of Efficient Stopping Criteria for Turbo Decoding**." Journal of Physics: Conference Series. Vol. 1804. No. 1. IOP Publishing, 2021.

[10] A. R. Hussien, "**Universal Decoder for Low Density Parity Check, Turbo and Convolutional Codes**, "Ph. D. thesis, Laval Quebec University, 2011.

[11] A. Jemibewon, "**A Smart Implementation of Turbo Decoding for Improved Power Efficiency**, " Msc. thesis, Virginia Polytechnic Institute and State University, 2000.

[12] L. Yun, A.Hashemi, J. Cioffi, and A. Goldsmith "**Construction of polar codes with reinforcement learning**." *IEEE Transactions on Communications* 70.1 (2021): 185-198.

[13] R.Tobias, and J. Sloof. "**Implementation and evaluation of Polar Codes in 5G**." (2019).

[14] 3GPP Technical Specifications 36.212, "**Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (EUTRA); Multiplexing and channel coding (Release 15)**," 2019.

[15] F. U. Ullah. " **Polar code design and decoding for magnetic recording**. " Diss. Georgia Institute of Technology, 2014.

[16] C.Leroux, A. J. Raymond, G. Sarkis, et al., "**A semi-parallel successive-cancellation decoder for polar codes**," IEEE Transactions on Signal Processing, Vol. 61, No. 2, 289–299, 2012.

[17] Y. Bo, and K. Parhi. "**Low-latency successive-cancellation polar decoder architectures using 2-bit decoding**." IEEE Transactions on Circuits and Systems I: Regular Papers 61.4 (2013): 1241-1254.

[18] P. Alptekin, and E. Arıkan. "**A two phase successive cancellation decoder architecture for polar codes**." *2013 IEEE International Symposium on Information Theory*. IEEE, 2013.

[19] G. Pascal, G. Sarkis, C. Thibeault, and W. J. Gross. "**A 638 Mbps low-complexity rate 1/2 polar decoder on FPGAs.**" 2015 IEEE Workshop on Signal Processing Systems (SiPS). IEEE, 2015.

[20] D. Onur, and E. Arıkan. "**A high-throughput energy-efficient implementation of successive cancellation decoder for polar codes using combinational logic**." IEEE Transactions on Circuits and Systems I: Regular Papers 63.3 (2016): 436-447.

[21] H. Yujie. "**The Hardware Acceleration of SC Decoder for Polar Code towards HLS Optimization**." *2018 International SoC Design Conference (ISOCC)*. IEEE, 2018.

[22] C. Yi. "**An Improved Successive-Cancellation Decoding Algorithm for Polar Code Based on FPGA**." *2018 Progress in Electromagnetics Research Symposium (PIERS-Toyama)*. IEEE, 2018.

[23] D. Yann. "**Model-based Design of Hardware SC Polar Decoders for FPGAs**." *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 13.2 (2020): 1-27.

[24] S. E. Anderson. **Applications of Algebraic Geometric Codes to Polar Coding** Clemson University. 5-2015

[25] K. El-Abbasy. "**Polar codes Bhattacharyya parameter generalisation**." IET Communications 14.11 (2020): 1718-1729.

[26] L. Wang. "**The method to recognize linear block code based on the distribution of code weight**." 2018 10th International Conference on Communication Software and Networks (ICCSN). IEEE, 2018.

[27] V. Harish, V. Emanuele and Y. HONG. "**A comparative study of polar code constructions for the AWGN channel**. "arXiv preprint arXiv:1501.02473, 2015

[28] P. Alptekin. "**An FPGA implementation architecture for decoding of polar codes**." 2011 8th International symposium on wireless communication systems. IEEE, 2011.

[29] N. Hussami, S. B. Korada, and R. Urbanke, "**Performance of polar codes for channel and source coding**, " in Proc. 2009 IEEE Int. Symp. Inform. Theory, (Seoul, South Korea), pp. 1488 1492, 28 June - 3 July 2009.

[30] S. B. Korada, **Polar codes for channel and source coding**. PhD thesis, EPFL, Lausanne, 2009.

[31] E. Sasoglu, E. Telatar and E. Yeh, **Polar codes for the two-user multipleaccess channel**, Proceedings of the 2010 IEEE Information Theory Workshop, Cairo, Egypt, January 6-8, 2010.

[32] E. Abbe and E. Telatar, MA C polar codes and matroids, **Information Theory and Applications Workshop**, ITA 2010, pp.1-8, Jan. 31 - Feb. 5, 2010.

[33] E. Hof, I. Sason and S. Shamai, "**Polar coding for reliable communications over parallel channels**", Proceedings of the 2010 IEEE Information Theory Workshop, Dublin, Ireland, Aug. 30 - Sept. 3, 2010.

[34] H. Mahdavifar and A. Vardy, " **Achieving the secrecy capacity of wiretap channels using polar codes,** " to appear in the IEEE Trans. Inform. Theory, vol. 57, 2011. (Also available on arXiv:1007.3568.)

[35] V. Harish, V. Emanuele and Y. HONG. "**A comparative study of polar code constructions for the AWGN channel**." arXiv preprint arXiv:1501.02473 (2015).

[36] G. Sarkis. **Efficient Encoders and Decoders for Polar Codes: Algorithms and Implementations** Department of Electrical and Computer Engineering McGill University Montreal, Canada April 2016.

[37] S. Boaz, and I. Tal. "**A lower bound on the probability of error of polar codes over BMS channels**." IEEE Transactions on Information Theory 65.4 (2018): 2021-2045

[38] L. Waslon TA, Y. Xue, and E. S. Sousa. "**Performance of modulation diversity with polar encoding in Rayleigh fading channel**." Proceedings of the 16th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks. 2019.

[39] P. Ramtin. Polar codes: **Construction and performance analysis**. Diss. 2011.

[40] H. Mahdavifar, M. El-Khamy, J. Lee and I. Kang, "**Performance Limits and Practical Decoding of Interleaved Reed-Solomon Polar Concatenated Codes**," in IEEE Transactions on Communications, vol. 62, no. 5, pp. 1406-1417, May 2014, doi: 10.1109/TCOMM.2014.050714.130602.

[41] M. Amiridi, "**Polar-code construction and decoding techniques**", Diploma Work, School of Electrical and Computer Engineering, Technical University of Crete, Chania, Greece, 2018 https://doi.org/10.26233/heallink.tuc.72291

[42] S. Li, M. Cai, R.Edwards, Y.Sun, and L. Jin, (2022). **Research on encoding and decoding of non-binary polar codes over GF** (2m). Digital Communications and Networks.

[43] R.Sornalatha , A.Muthumanicckam, L.Vijayprabakaran. **FPGA Implementation of Successive Cancellation Polar Decoder Architecture.** 2015.

[44] L. Gal, Bertrand, C. Leroux, and C. Jego. "**High-performance software implementations of SCAN decoder for polar codes**." Annals of Telecommunications 73.5 (2018): 401-412.

[45] C. Fengyi. "**Bit-flip algorithm for successive cancellation list decoder of polar codes**." IEEE Access 7 (2019): 58346-58352.

[46] X. Weihong. "**Deep learning-aided belief propagation decoder for polar codes**." IEEE Journal on Emerging and Selected Topics in Circuits and Systems 10.2 (2020): 189-203.

[47] C. Zhang, B. Yuan, and K. K. Parhi, "**Reduced-latency SC polar decoder architectures**," in Proc. Int. Conf. Commun., June 2012, pp. 3471–3475.

[48] C. Zhang, and K. Parhi "**Low-latency sequential and overlapped architectures for successive cancellation polar decoder**, " IEEE Trans. Signal Processing, vol. 61, no. 10, pp. 2429–2441, May 2013

[49] Y. Wang, , Q. Wang, Y. Zhang, S. Qiu, & Z. Xing. "**An area-efficient hybrid polar decoder with pipelined architecture**." IEEE Access 8 (2020): 68068-68082.

[50] W. Alebady and A. Hamad "**Turbo polar code based on soft-cancelation algorithm,**" Indonesian Journal of Electrical Engineering and Computer Science, Vol. 26, No. 1, pp. 521~530 April 2022.

[51] A. Hashemi. "**Memory-efficient polar decoders**." IEEE Journal on Emerging and Selected Topics in Circuits and Systems 7.4 (2017): 604-615.

[52] G. Sarkis. **Efficient Encoders and Decoders for Polar Codes: Algorithms and Implementations** Department of Electrical and Computer Engineering McGill University Montreal, Canada April 2016.

[53] S. Hauck and A. DeHon, "**Reconfigurable Computing The Theory and Practice of FPGA-Based Computation**," Morgan Kaufmann is an imprint of Elsevier, 2008.

[54] J. R. Hess, "**Implementation of a Turbo Decoder on a Configurable Computing Platform**," Msc. thesis, Virginia Polytechnic Institute and State University, 1999.

[55] M. S. Jasam, and H. F. Abdulsada. "**FPGA Implementation of 3 bits BCH Error Correcting Codes**." International Journal of Computer Applications 71.7 (2013): 35.

[56] S. Kilts, "**Advanced FPGA Design Architecture, Implementation, and Optimization**," A John Wiley & Sons, Inc., 2007.

[57] S. M. Sreedaranath, "**Rapid Prototyping of Software Defined Radios using Model Based Design for FPGAs**," Msc. thesis,Virginia Polytechnic Institute and State University, July 22, 2010.

[58] J. P. Deschamps, G.J. Antoine Bioul and G. D. Sutter, "**Synthesis of arithmetic circuits FPGA, ASIC and Embedded Systems**," A John Wiley & Sons, Inc., 2006.

[59] S. Churiwala, "**Designing with Xilinx FPGAs using Vivado**," (eBook)DOI, Library of Congress Control Number: 2016951983, Springer International Publishing Switzerland 2017.

[60] J. Peter. "**Analyzing a Low-bit rate Audio Codec-Codec2-on an FPGA**." 2021 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, 2021.

[61] H. Rodrigues, and F. Dias. "**Hardware Implementation of an Artificial Neural Network with an embedded microprocessor in a FPGA**." 8th International Conference and Workshop on Ambient Intelligence and Embedded Systems", Funchal. 2009.

[62] O. Arcas-Abella, G. Ndu, N. Sonmez, M. Ghasempour, A. Armejach, J. Navaridas, W. Song, J. Mawer, A. Cristal, and M. Lujan, "**An empirical evaluation of high-level synthesis languages and tools for database acceleration**," in 24th International Conference on Field Programmable Logic and Applications (FPL), September 2014.

[63] G. Konstantinos."**An evaluation of vivado HLS for efficient system design**." 2016 International Symposium ELMAR. IEEE, 2016.

[64] R. de Bulnes and Y. Maldonado. "**Comparison of multi-objective evolutionary algorithms for high level synthesis in FPGA devices**." Computacióny Sistemas 22.2 (2018): 425-437.

[65] Xilinx Inc., **Vivado Design Suite User Guide** v2015.1, 2015.

[66] "**Vivado Design Suite User Guide: High-Level Synthesis**," UG902 (v2018.3) December 20, 2018.

[67] D. Lee, D. Villasenor, W. Luk, and W. Leong, "**A Hardware Gaussian Noise Generator Using the Box-Muller Method and Its Error Analysis**," IEEE Transactions on Computers, vol. 55, no. 6, June 2006.

[68] M. Engels, "**Wireless OFDM Systems - How to Make Them Work**," Kluwer Academic, 2002.

Appendix A

Equation of Single Parity-Check Codes and Repetition Codes

For Single Parity-Check Codes:

$$\Pr(c_i \neq \hat{c}_i) = 1 - \Pr(c_i = \hat{c}_i)$$

$$= 1 - \sum_y p(c_i = \hat{c}_i, y)$$

$$= 1 - \sum_y p(c_i = \hat{c}_i \mid y)\, p(y)$$

$$\geq 1 - \sum_y p(y) \max_{\hat{c}_i \in \{0,1\}} p(c_i = \hat{c}_i \mid y) \qquad (\mathbf{A}.1)$$

Equation **(**A.1) shows that the decoder that minimizes the error probability is the one that maximizes the probability p (c_i |y) for all c_i . We call this probability the a posterior probability (APP), the decoder the maximum a posterior (MAP) decoder, and the decision the MAP decision.

The MAP decision $\hat{c}_i$ is given by

$$\hat{c}_i(\mathbf{y}) = \arg\max_{c_i} p(c_i \mid \mathbf{y})$$

$$= \arg\max_{c_i} \frac{p(\mathbf{y}|c_i)p(c_i)}{p(\mathbf{y})} \qquad \text{(Bayes' Rules)}$$

$$= \arg\max_{c_i} p(\mathbf{y} \mid c_i)p(c_i) \qquad (A.2)$$

where in the last step, we omitted $p(\mathbf{y})$, as it does not depend on c_i and does not take part in the optimization process.

Suppose we are interested in detecting bit c_2. We calculate the MAP decision for c_2 by first computing the objective function in (A.2) for both the values c_2

$\in \{0, 1\}$ and then finding the value of c_i that maximizes it. We start by computing $p(\mathrm{y}|c_i{=}0)\ \Pr(c_i{=}0)$ as follows:

$$p(\mathbf{y} \mid c_2 = 0)\ \Pr(c_2 = 0) = \underbrace{p(y_0, y_1 \mid c_2 = 0)}_{\text{from } y_0, y_1}\ \underbrace{p(y_2 \mid c_2 = 0)}_{\text{from } y_2}\ \underbrace{\Pr(c_2 = 0)}_{\text{Message Source}}. \qquad \text{(A.3)}$$

The right-hand side of (A.3) depends on the following:

1. $p(y_0, y_1 \mid c_2 = 0)$: This is the likelihood of c_2 being zero given noisy observations y_0 and y_1 of c_0 and c_1, respectively. This term carries the information that y_0 and y_1 convey about c_2.

2. $p(y_2 \mid c_2 = 0)$: This is the likelihood of c_2 being zero given noisy observation y2. This term carries the information that y_2 conveys about c_2.

3. Pr $(c_2 = 0)$: Assuming m_0 and m_1 are equally likely to be zero or one, c_2 is equally likely to be zero or one as well, and Pr $(c_2 = 0) = 1/2$.

Expanding $p(y_0, y_1 \mid c_2 = 0)$ term on the right-hand side of (A.3), we get

$$\begin{aligned} p(y_0, y_1 \mid c_2 = 0) &= \sum_{c_0, c_1} p(y_0, y_1, c_0, c_1 \mid c_2 = 0) \\ &= \sum_{c_0, c_1} p(y_0, y_1 \mid c_0, c_1, c_2 = 0) p(c_0, c_1 \mid c_2 = 0) \\ &= p(y_0, y_1 \mid c_0 = 0, c_1 = 0, c_2 = 0) \Pr(c_0 = 0, c_1 = 0 \mid c_2 = 0) \\ &\quad + p(y_0, y_1 \mid c_0 = 1, c_1 = 0, c_2 = 0)\ \Pr(c_0 = 1, c_1 = 0 \mid c_2 = 0) \\ &\quad + p(y_0, y_1 \mid c_0 = 0, c_1 = 1, c_2 = 0)\ \Pr(c_0 = 0, c_1 = 1 \mid c_2 = 0) \\ &\quad + p\big(y_0, y_1 \mid c_0 = 1, c_1 = 1, c_{2,} = 0\big)\ \Pr(c_0 = 1, c_1 = 1 \mid c_2 = 0) \end{aligned} \qquad \text{(A.4)}$$

Since c_0 and c_1 must be identical when c_2 = 0, the middle two terms are zero, and (A.4) simplifies to:

$$\mathrm{p}(y_0, y_1 \mid c_2 = 0) = \mathrm{p}(y_0, y_1 \mid c_0 = 0, c_1 = 0, c_2 = 0)\Pr\,(c_0 = 0, c_1 = 0 \mid c_2 = 0)$$
$$+\mathrm{p}(y_0, y_1 \mid c_0 = 1, c_1 = 1, c_2 = 0)\,\Pr(c_0 = 1, c_1 = 1 \mid c_2 = 0) \quad \text{(A.5)}$$

But since the message source is independently and uniformly distributed, we have

$\Pr\,(c_0=1, c_1=1|c_2=0) = \Pr\,(m_0=1, m_1=1|c_2=0) = 1/2$

Therefore, $p(y_0, y_1 \mid c_2 = 0)$in (A.5) is simplified to

$$p(y_0, y_1 \mid c_2 = 0) = \frac{1}{2}\big(p(y_0, y_1 \mid c_0 = 0, c_1 = 0, c_2 = 0) + p(y_0, y_1 \mid c_0 = 1, c_1 = 1, c_2 = 0)\big)$$
$$= \frac{1}{2}\big(p(y_0 \mid c_0 = 0)p(y_1 \mid c_1 = 0) + p(y_0 \mid c_0 = 1)p(y_1 \mid c_1 = 1)\big) \quad (A.6)$$

Using this value of $p(y_0, y_1 \mid c_2 = 0)$ in (A.3), we get

$$p(\mathbf{y} \mid c_2 = 0)\,\Pr(c_2 = 0) = \tfrac{1}{4}\big(p(y_0 \mid c_0 = 0)p(y_1 \mid c_1 = 0) + p(y_0 \mid c_0 = 1)p(y_1 \mid c_1 = 1)\big)p(y_2 \mid c_2 = 0) \quad (A.7)$$

Where once again, we have used the fact that the source is uniformly distributed, and $\Pr\,(c_2 = 0) = 1/2$. By following the same procedure, we used for $p(\mathbf{y} \mid c_2 = 0)\Pr\,(c_2 = 0)$, we compute

$$\mathrm{p}(\mathbf{y} \mid c_2 = 1)\,\Pr(c_2 = 1) = 1/4\,(p(y_0|c_0=0)\,p(y_1|c_1=1)+p(y_0|c_0=1)p(y_1|c_1=0))p(y_2|c_2=1) \quad \text{(A.8)}$$

The last step of the decoder is to use these two probabilities to make a decision about i-th bit, according to

$$p(c_i = 0 \mid \mathbf{y}) \underset{1}{\overset{0}{\gtrless}} p(c_i = 1 \mid \mathbf{y})$$

or

$$p(\mathbf{y} \mid c_i = 0)\,\Pr(c_i = 0) \underset{1}{\overset{0}{\gtrless}} p(\mathbf{y} \mid c_i = 1)\,\Pr(c_i = 1)$$

In simple words, it means that decide the bit is zero if

$p(y|c_i=0)\ \Pr\ (c_i=0)>p(y|c_i=1)\ \Pr\ (c_i=1)$

And one otherwise. We can further simplify it by converting it into a ratio and taking the natural logarithm of both sides as follows:

$$\frac{p(\mathbf{y} \mid c_i = 0)\Pr(c_i = 0)}{p(\mathbf{y} \mid c_i = 1)\Pr(c_i = 1)} \underset{1}{\overset{0}{\gtrless}} 1$$

$$\log\left(\frac{p(\mathbf{y}|c_i=0)\Pr\,(c_i=0)}{p(\mathbf{y}|c_i=1)\Pr\,(c_i=1)}\right) \underset{1}{\overset{0}{\gtrless}} 0 \qquad \text{(A.9)}$$

The side on the left of (A.9) can further be simplified as

$$\log\left(\frac{p(\mathbf{y} \mid c_i = 0)\,\Pr(c_i = 0)}{p(\mathbf{y} \mid c_i = 1)\,\Pr(c_i = 1)}\right) = \log\left(\frac{p(y_0 \mid c_0 = 0)p(y_1 \mid c_1 = 0) + p(y_0 \mid c_0 = 1)p(y_1 \mid c_1 = 1)}{p(y_0 \mid c_0 = 0)p(y_1 \mid c_1 = 1) + p(y_0 \mid c_0 = 1)p(y_1 \mid c_1 = 0)} \times \frac{p(y_2 \mid c_2 = 0)}{p(y_2 \mid c_2 = 1)}\right)$$

$$= \log\left(\frac{\frac{p(y_0 \mid c_0 = 0)}{p(y_0 \mid c_0 = 1)}\frac{p(y_1 \mid c_1 = 0)}{p(y_1 \mid c_1 = 1)} + 1}{\frac{p(y_0 \mid c_0 = 0)}{p(y_0 \mid c_0 = 1)} + \frac{p(y_1 \mid c_1 = 0)}{p(y_1 \mid c_1 = 1)}} \times \frac{p(y_2 \mid c_2 = 0)}{p(y_2 \mid c_2 = 1)}\right)$$

$$= \log\left(\frac{(\ell_0\ell_1+1)\ell_2}{\ell_0+\ell_1}\right) \tag{A.10}$$

Where

$$\ell_j = \frac{p(y_j \mid c_j = 0)}{p(y_i \mid c_i = 1)}$$

We can use trigonometric identities to reduce (A.10) to

$$\log\left(\frac{p(c_2 = 0 \mid \mathbf{y})}{p(c_2 = 1 \mid \mathbf{y})}\right) = \underbrace{2\tanh^{-1}\left(\tanh\left(\frac{L_0^{(i)}}{2}\right)\tanh\left(\frac{L_1^{(i)}}{2}\right)\right)}_{\text{Extrinsic LLR}} + \underbrace{L_2^{(i)}}_{\text{Intrinsic LLR}}$$

$$= L_2^{(e)} + L_2^{(i)} \tag{A.11}$$

where

$$L_j^{(i)} = \log\left(\frac{p(y_j \mid c_j = 0)}{p(y_j \mid c_j = 1)}\right)$$

For repetition code:

$$\log\left(\frac{p(c_k=0|\mathbf{y})}{p(c_k=1|\mathbf{y})}\right) = \underbrace{\sum_{j\neq k} L_j^{(i)}}_{\text{Extrinsic LLR}} + \underbrace{L_k^{(i)}}_{\text{Intrinsic LLR}} \tag{A.12}$$

Appendix B

Algorithm B.1: Hardware implementation of Polar code using MicroBlaze in SDK

Initialization: Define: Pseudo_random: function to generate a binary data. Pencoder: function of polar code encoding. Geng: function to generate a gaussian data. polar_SC_decode: function of polar code decoding.

Procedure:
main:
Define: ber,fer,errs,nerrs,i,NN,KK,LEN,FN, EbNodB, Eb_No
max_trans_frames, min_frame_num , min_frame_errors, nframe
for i=0< 2*LEN do:
transmittedData[i] = 2 * (encodedData[i] - 0.5)
end-of (for)
for i=0 <2*LEN do
noisyData[i] = transmittedData[i] + noise[i]
end-of (for)
for i=0<2*LEN do
llr_input[i] = -2.0 * noisyData[i] * sqrt(ebn0_num)
end-of (for)
Eb_No = pow(10,(design_SNR/10.0))
Initialization system's functions: Pseudo_random, Encoder,Geng,Decoder
errs=0
nferr=0

```
while nframe <= maxnoframe do:
Set_data
Set_datalen
XPseudo_random_Start
Set_genu
Set_geny
Pencode_Start
mytimer.startTimer()
time_Encoder= mytimer.getElapsedTimerInSeconds();
for(int i=0 < 2*LEN do:
transmittedData[i] = 2 * (encodedData[i] - 0.5)
end of for
Set_noise
Set_sigma2
Set_noiseLen
XGeng_Start
for (int i=0 <2*LEN do:
noisyData[i] = transmittedData[i] + noise[i]
end of for
Set_geninitial_llr
Set_genB
Polar_sc_decode_Start
mytimer.startTimer()
time_Decoder= mytimer.getElapsedTimerInSeconds( )
printf("Decoder elapsed time: %f\n", time_Decoder);
err = 0
```

```
for (int i=0 <LEN do:
if (decodedData[i] != data[i])do
err++
end of if
errs +=err
end of for
if (err>0)do:
nferr++
end of if
if nframe%FN==0 do:
ber = errs/nframe/LEN;
fer = nferr/nframe;
end of (if)
printf(time_Encoder, time_Decoder);
printf(EbNodB, ber,fer,nframe, time_Decoder);
nframe++;
if ((loop+1) >= min_frame_num && nferr >= min_frame_errors) do:
break;
end of (if)
return 0
end-of (for)
```

الخلاصة

نظام الاتصالات هو نظام نموذجي يحدد التبادل المعقد بين محطتين محوريتين: المرسل والمستقبل. عندما تمر الإشارات أو المعلومات من مصدرها إلى وجهتها النهائية ، فإنها تتنقل في قناة معقدة مشبعة بالعديد من الانحرافات ، مثل الضوضاء والتوهين والتشويه. لذلك ، يتم استخدام أكواد تصحيح الأخطاء للتحكم في الأخطاء في البيانات عند إرسالها عبر قناة اتصال غير موثوقة أو مزعجة.

تمثل الرموز القطبية فئة ناشئة من أكواد تصحيح الأخطاء مع القدرة على الاقتراب من سعة قناة منفصلة بلا ذاكرة. تم اختيار الكود القطبي لقناة التحكم في الجيل الخامس، الراديو الجديد (NR)، لأنه يمكن أن يصلح الأخطاء بشكل جيد حتى في الحالات التي تكون فيها نسبة القدرة من الإشارة إلى الضوضاء منخفضة. لقد أثبت أن الأداء يمكن أن يقترب من حد شانون لأطوال المقاطع الكبيره. وتم اختياره في مشروع شراكة الجيل الثالث (3GPP) كمخطط ترميز القنوات لقنوات التحكم في النطاق العريض المتنقل المحسن(EMBB).

تقترح هذه الدراسة العمل تنفيذ FPGA للترميز القطبي والذي يستخدم خوارزمية الإلغاء المتتالي (SC). مفكك الشفرة SC هو مفكك الشفرة التقليدي للشفرة القطبية التي يكون مدخلها هو نسبة احتمالية لوغاريتمية (LLR) ومخرجاتها (ثنائية). وتم تحسين خوارزمية فك الترميز SC الخاصة بـ Polar Code و تطويرها لجعلها أكثر عملية. لقد ثبت أن استخدام عدد أقل من البتات لتمثيل قيم احتمالية السجل (LLR) في وحدة فك التشفير له تأثير كبير على مساحة السيليكون المستخدمة والإنتاجية دون تأثير كبير على الأداء.

بالإضافة إلى أنواع البيانات القياسية الأصلية لـ C ++ التي تم إنشاؤها على حدود 8 بت ، يوفر Vivado High-level synthesis (HLS) أنواعًا دقيقة من البتات و التي تسمح بتحديد أي عرض عشوائي للبتات. في ضوء ذلك ، فإنه يوفر أيضًا مقارنة لبناء وحدة فك الترميز القطبية لل (SC) باستخدام أنواع مختلفة من البيانات الثابتة ذات الدقة عشوائيه بدلاً من تمثيل float الأكثر تكلفة لتقليل وقت المعالجة والمساحة المستخدمة.

لقد استخدمت هذه الدراسة جهاز FPGA Kintex-7 ذو رقم (XC7K325T-2FFG900C) Xilinx مع توجيهات التوازي المختلفة لضمان الحصول على فاصل البدء المستهدف ومنطقة

السيليكون. أصبح هذا ممكنًا بفضل درجة المرونة العالية لجهاز FPGA في تصميم وتنفيذ أنظمة النماذج الأولية. وتم استخدام الموجهات المختلفة التي يوفرها HLS ، مثل العمل بالتوازي ، وفك الانقسام ، وتقسيم المصفوفه ، لتحقيق مفاضلة بين الأداء (التاخير وسرعة المعالجه) ومساحة السيليكون. لقد ثبت أن استخدام التوجيهات ليس له أي تأثير على أداء BER ولكنه يقلل من وقت فك الترميز.

وتم إجراء اختبارات محاكاة مختلفة على قناة الضوضاء الغوسية البيضاء المضافة (AWGN) لإثبات فعالية النظام المقترح.وأظهرت النتائج انخفاضًا في زمن الانتقال بحوالي 76٪ وانخفاض ملحوظ في العناصر المنطقية ووحدات DSP (DSP48E) بحوالي 50٪ و 86٪ على التوالي. ويظهر تنفيذ الأجهزة(hardware) لانظمة الترميز القطبي المختلفة بدقة عشوائية بدلاً من نوع البيانات float تدهور سطحي في الأداء.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابـل
كلية الهندسة / قسم الهندسة الكهربائية

تقليل مصادر مصفوفة البوابات المنطقية القابلة للبرمجة موقعيا الخاصه بالمرمز القطبي باستخدام خوارزمية الالغاء المتتالي

رســـالـــــة

مقدمة الى كلية الهندسة في جامعة بابل

كجزء من متطلبات نيل درجة الماجستير

في الهندسة / الهندسة الكهربائية / الاتصالات

من قبل

زبيده مهدي شمير خضير

أشــراف

الأستاذ الدكتور / احمد عبد الكاظم حمد

١٤٤٤هـ ٢٠٢٣م